

LOS MEJORES LISTADOS PARA TU MICRO

EDITA: 

-año I-Nº 2

PVP 200 Pts. IVA Incl.

SONIMAG'87
NOVEDADES SONY

La instrucción Sound al descubierto

LISTADOS
GUSANO GLOTON
CUBILETES
CAP INTERMITENTE
PUENTES
CUCO
FICHERO DE PANTALLAS
DRIVER

STOP BALL

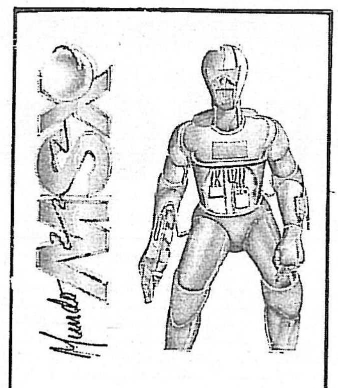
COSA NOSTRA

FERNANDO MARTIN

PVP CANARIAS 200 Pts.



¡Tu cinta para grabar guay!
HOLIDAY
HOLIDAY C-40
HOLIDAY C-90
Tu cinta virgen de 40, 60 y 90
HOLIDAY C-90
V. CARRERA
Fabricada por IBEROFON, S.A.
Avenida de Fuentemar, 35 - Polígono Industrial de Coslada - MADRID
Teléfonos 671 22 00-04-08-12 - Télex 42797 FONO E - Telefax (91) 671 39 09
Direct Metal DMM Mastering
COMPACT disc DIGITAL AUDIO
discoflex®

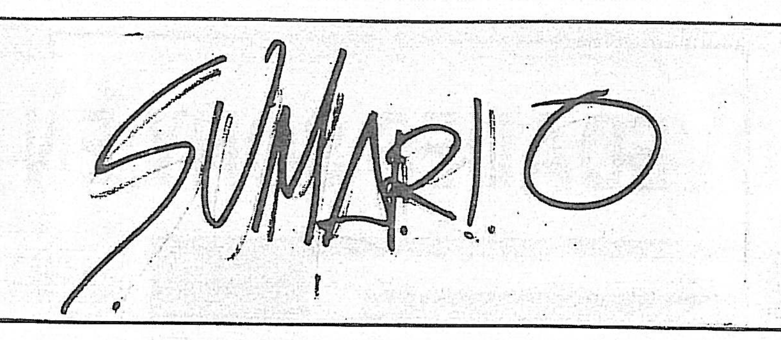


Director:
Angel Herrero Fernández
Director Técnico:
Luis Sanguino
Coordinador Editorial:
Félix Santamaría
Software:
Angel García
Secretaria Redacción:
Mercedes Matons
Diseño e Ilustraciones:
Javier Caballero
Colaboradores:
Antonio García
A. Gustavo Chico
Andrés M. García

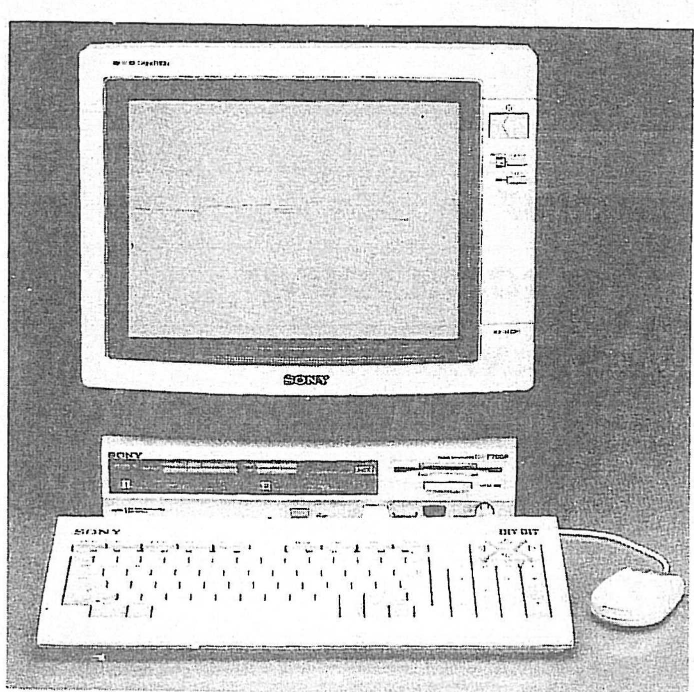
Mundo MSX es una publicación del Grupo Editorial SYGRAN, S.A.
Polig. Ind. Valdonaire.
C/Apolonio Hernández.
HUMANES (Madrid).
Director Adjunto:
Andrés Franco

Publicidad y suscripciones:
GENESIS
Tomás López, 3-6°
28009 Madrid
Tel. 401 77 54

Fotocomposición:
Speed Letra, S.A.
Imprime:
Gráficas Osiris, S.A.
Brañuelas, 29
Fuenlabrada
Distribuye:
G.M.E. Pza. Castilla, 3.
Madrid
Depósito Legal:
M-31674-1987



NOVEDADES SONY EN EL SONIMAG'87



HB-F700S

Procesador	Z80A
Frecuencia Clock	3 58 MHZ
Memoria ROM	32 KBytes (BASIC MSX Versión 1.0)
Memoria RAM	80 KBytes
	64 KBytes (memoria principal)
	16 KBytes (memoria de video)
Texto en pantalla	40 caracteres × 24 lineas
Resolución Gráfica	SCREEN 2 256 × 192 puntos 16 colores
	SCREEN 3. 64 × 48 bloques 16 colores
Colores:	16
Teclado	Profesional (tipo QWERTY). (incluye Ñ)
	75 teclas
	5 teclas de función (10 funciones)
Sonido	8 Octavas/3 voces
Conexión Cartuchos:	2 conectores de 50 contactos.

Conexiones Directas	— Televisor.
	— Cassette
	— Interface RS-232
	— Impresora
	— Joystick (2)
	— Lector de Diskettes MSX
	— Bola gráfica
Interface Cassette:	Velocidad variable —1.200/2.400 Baudios
Gráficos Sprite.	32 planos
Lenguajes.	FORTRAN. COBOL. PASCAL. «C». LOGO y ENSAMBLADOR.
Accesorios Suministrados	— Caja de accesorios que incluye: • Alimentador AC-220V • Cable conexión cassette • Manual «Aprenda a programar en BASIC MSX»

Especificaciones HB-20P

SONY sigue manteniendo un fuerte compromiso con el sistema MSX como lo demuestra el hecho de su presencia en el SONIMAG-87 con un amplio STAND en el que figuraban todos los ordenadores, periféricos y programas que la multinacional japonesa sigue lanzando al mercado.

En el Salón del Sonido la Imagen y la Electrónica, celebrado en Barcelona entre el 28 de octubre al 4 de septiembre, la informática no tenía una representación muy nutrida, y las pocas firmas de este sector que acudieron al certamen (con la excepción de AMSTRAD) no presentaban muchas innovaciones, sin embargo SONY sí ha dado un paso importante de cara a revitalizar un sistema de ordenadores que tanta aceptación ha tenido en nuestro país. Los equipos MSX presentados por SONY en el SONIMAG-87 han ido respaldados por algo que siempre hemos hechado de menos los usuarios de éste sistema como son las utilidades, aplicaciones y demás herramientas útiles para trabajar con un equipo de grandes posibilidades potenciales, pero que hasta ahora no había sido desarrollado suficientemente.

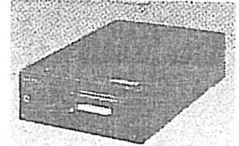
En éste SONIMAG-87, SONY ha roto una lanza en este sentido y ha presentado un amplio catálogo de 72 páginas en el que figura toda la gama de productos MSX que dicha firma comercializa en la actualidad, tanto en el terreno del software como del hardware.

En éste catálogo, ordenados por materias, figuran programas de educación en cartucho y cinta, lenguajes de programación, material didáctico, aplicaciones, ges-

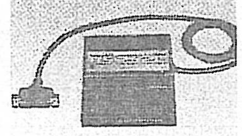
HB-20P



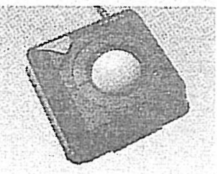
GAMA DE PERIFERICOS SONY MSX



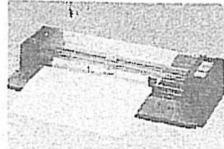
HBD-30W.
Lector de diskettes de una cara (500 KBytes) y de doble cara (1 Mbyte) Puede utilizarse como primera unidad, conectándose al ordenador mediante el cable de interface HBK-30 o bien como segunda unidad, utilizando el cable HBK-35



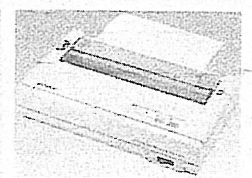
HBI-232.
Cartucho de comunicaciones Permite conectar su HIT BIT con cualquier ordenador equipado con RS-232C via telefónica mediane un modem o accplador acustico o bien directamente utilizando un cable de extensión



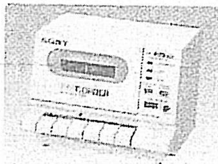
GB-6.
Nueva bola gráfica de reducidas dimensiones especialmente indicada en programas que utilizan «iconos» menús de selección gráfica donde es indispensable un movimiento rápido y preciso del cursor por la pantalla



PRN-C41.
La plotter se distingue de una impresora normal por su capacidad de dibujar gráficos y por su peculiar diseño Resulta ideal para diseñadores, arquitectos o estudiantes por sus cualidades en el diseño de estadísticas, gráficas y listados a color, gracias a sus cargas de tinta de color negro, verde, azul y rojo. Funciona con papel de cualquier tamaño y dispone de 15 tamaños distintos de letra.



PRN-M09.
Nueva impresora MSX de SONY con una velocidad de impresión de 75 caracteres/sg. Dispone de varios tipos de letra, pudiéndolos imprimir en alta calidad (NLQ). Sus muchas prestaciones, reducido tamaño y diseño atractivo la configuran como impresora ideal para la oficina y trabajos profesionales (cartas, informes, listados, etc.)



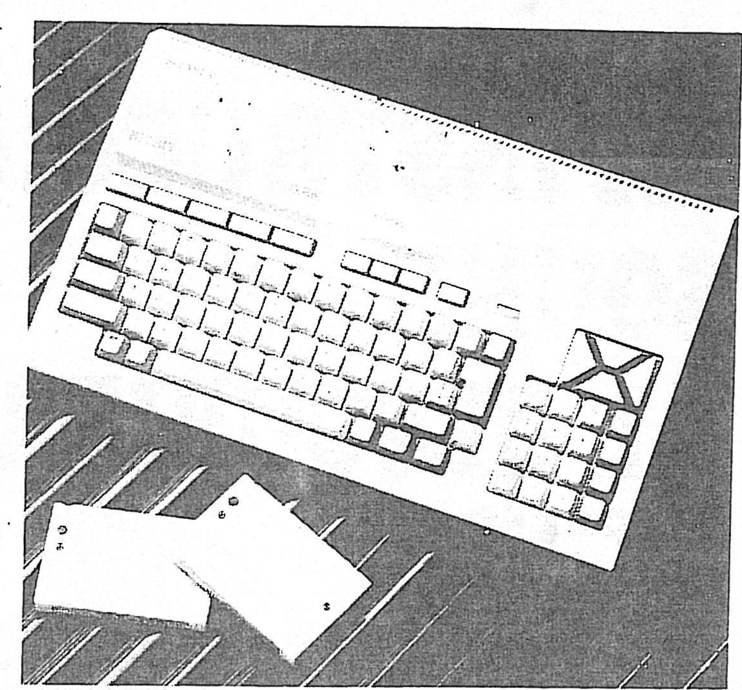
SDC-600S.
Nuevo grabador/reproductor de cassettes de alta velocidad especialmente diseñado para ordenadores MSX Dispone de dos velocidades de avance de cinta, pudiendo reducir a la mitad el tiempo de carga de los programas Incorpora un selector de fase para evitar los problemas de carga

ción, juegos y entretenimientos, libros, revistas, etc., así como toda la gama de ordenadores y periféricos disponibles en el actual mercado nacional pertenecientes a la firma SONY.

Los ordenadores HIT-BIT más novedosos son básicamente tres, el HB-20P perteneciente a la primera versión del sistema MSX y los MSX 2, HB-F9S y HB F700S.

El primer equipo es una versión evolucionada de la ya clásica línea HIT-BIT, de reducido tamaño, presenta dos ranuras superiores para conexión de cartuchos, conexiones para monitor o TV, 2 salida para joysticks y conectores para cassette e impresora centronic, el resto de características de este aparato aparecen en el cuadro número 1.

El modelo HB-F9S es una variación más económica y compacta de los equipos HIT-BIT MSX 2 que como todos recordaréis aparecieron en el mercado incorporando por un lado el teclado independiente y por otro la unidad central con la diskettera y todas las conexiones del equipo.


HB-F9S





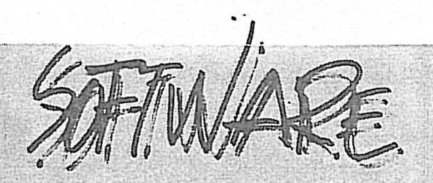
El HB-F9S es un equipo compacto que incorporan todos los circuitos en un único elemento al que puede acoplarse una unidad de disco externa, sus características completas aparecen en el cuadro número 2.

En lo alto de la gama tenemos el HB-F700S que obedece al concepto "clásico" de ordenador MSX 2 con teclado separado de la unidad central y que sale al mercado incorporando un ratón y un buen número de aplicaciones software que permite una inmediata utilización "profesional del equipo".

Especificaciones HB-F9S

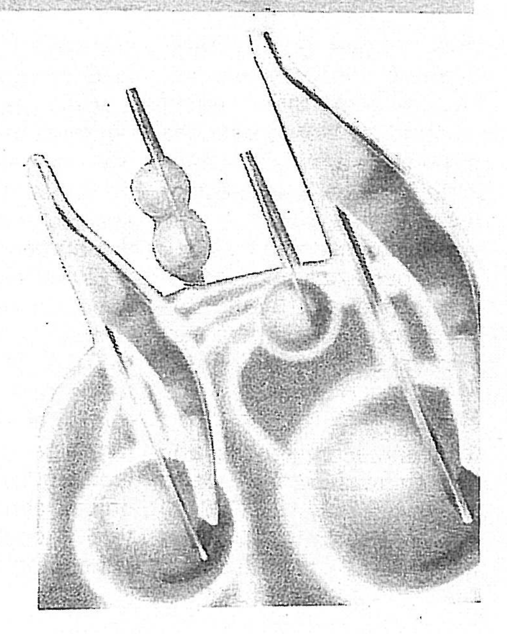
Procesador	Z80A
Frecuencia Clock	3 58 MHZ
Memoria ROM	96 KBytes
	48 KBytes (BASIC MSX Versión 2 0)
	48 KBytes Programa incorporado
Memoria RAM	256 KBytes
	128 KBytes (memoria principal)
	128 KBytes (memoria de video)
Texto en pantalla	40 caracteres × 24 líneas
	80 caracteres × 24 líneas
Resolución Gráfica	SCREEN 2 256 × 192 puntos 16 colores
	SCREEN 3 64 × 48 bloques 16 colores
	SCREEN 4 256 × 192 puntos 16 colores
	SCREEN 5 256 × 192 puntos 16 colores (BM)
	SCREEN 6 512 × 212 puntos 4 colores (BM)
	SCREEN 7 512 × 212 puntos 16 colores (BM)
	SCREEN 8 256 × 212 puntos 256 colores (BM)
Colores	16 colores de una paleta de 512
	256 colores simultáneos (SCREEN 8)
Teclado	90 teclas
	teclas de control 12
	teclas de función 5
	teclas de edición 8
	Teclado numérico independiente
Sonido	8 Octavas/3 voces

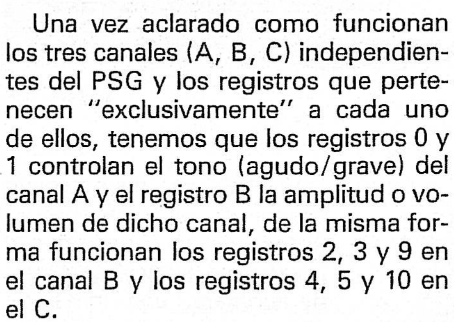
Conexión Cartuchos	2 conectores de 50 contactos
(BM) Bit Mapped Permite acceder a la pantalla punto a punto de forma que es posible mantener en cada punto un color distinto	
Conexiones Directas	– Cassette
	– RGB para monitor video
	– Video/Audio (color y monocromo)
	– RF para televisor
	– Impresora
	– Joystick (2)
	– Lector de Diskettes de 3 5 pulgadas
	– Bola gráfica, ratón, etc
Interface Cassette	Velocidad variable – 1 200/2 400 Baudios
Gráficos Sprite	32 planos Posibilidad de Multicolor
Reloj/Calendario	Con batería interna autónoma
Accesorios Suministrados	– Manual de instrucciones
	– Cable RF
	– Manual instrucciones del programa «Fichero Personal»
	– Manual «Introducción al BASIC MSX2»
	– «Manual de Referencia para programación en BASIC MSX2»
Software Suministrado	Incorporado en ROM «Fichero Personal» que incluye
	Listín telefónico
	Agenda Personal
	– Reloj calendario calculadora
Lenguajes Disponibles	FORTRAN COBOL PASCAL «C» LOGO y ENSAMBLADOR



LA INSTRUCCION SOUND AL DESCUBIERTO

2ª parte





Una vez aclarado como funcionan los tres canales (A, B, C) independientes del PSG y los registros que pertenecen "exclusivamente" a cada uno de ellos, tenemos que los registros 0 y 1 controlan el tono (agudo/grave) del canal A y el registro B la amplitud o volumen de dicho canal, de la misma forma funcionan los registros 2, 3 y 9 en el canal B y los registros 4, 5 y 10 en el C.

Vamos a profundizar más en el tema y a conocer cual es la misión del resto de los registros del PSG. Hasta ahora si hacemos sonar simultáneamente dos canales, introduciendo por ejemplo:

```
10 SOUND 8,15: SOUND 0,50
20 SOUND 9,15: SOUND 3,15
```

La única forma que conocemos de desactivar uno de ellos es poner a cero su registro de control de amplitud, es decir, introducir SOUND 8,0 o bien SOUND 9,0 según el canal que deseemos interrumpir. Pues bien el generador programable de sonido posee un registro que además de ser capaz de interrumpir la emisión de tonos de cada canal nos va a dar a conocer una posibilidad más de dicho generador.

Hemos visto o escuchado para ser más exactos, como un canal es capaz de emitir una nota, digamos limpia con una amplia diversidad de tonos. Pues bien esto es debido a que al conectar nuestro MSX el registro 7 del PSG activa únicamente la **salida de tono** de cada canal manteniendo desactivada la **salida de ruído** de cada uno de ellos. Veamos cual es la estructura de dicho registro y así entenderemos esto con mayor facilidad.

El registro siete posee 8 bits de los cuales en este artículo sólo vamos a tratar los 6 menos significativos ya que los dos restantes los utiliza el ordenador para controlar los joysticks y no intervienen para nada en la generación de sonidos. (Ver fig. 1).

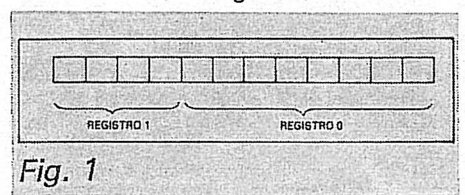


Fig. 1

EL REGISTRO 7

Decir que el resgistro siete activa únicamente las salidas de tono y no las de ruido de los tres canales al ser encendido nuestro ordenador, traducido a lenguaje informático es lo mismo que decir que en este momento los tres bits menos significativos de dicho registro, es decir, los situados más a la derecha del mismo toman valor cero mientras los tres restantes toman valor uno, como veís en este caso y al contrario que en otras ocasiones, el que un determinado bit tome valor cero significa que la salida que controla está activada y se desactivará cuando el valor de dicho bit sea igual a uno. De este modo el que el registro 7 mantenga activadas las salidas de tono, significa que su valor tendrá que ser: **111000.** Siempre que nos refiramos a este registro utilizaremos notación binaria, ya que así nos será mucho más sencillo saber qué salidas tenemos activadas. Por lo tanto si quisieramos reinicializar el registro 7 no tendríamos mas que introducir el mencionado valor de la siguiente forma: SOUND 7, & B111000.

Una vez explicada las funciones que cumplen los 6 bits del registro 7 vamos a ver cómo funcionan en la práctica con algunos ejemplos: comprobemos primero cómo suena un canal cuando activamos la salida de ruido.

En el pequeño programa anterior en el que hacíamos que dos canales emitiesen simultáneamente dos tonos diferentes vamos a introducir una nueva línea con el siguiente formato:

```
30 SOUND 7, &B111000
```

Si ejecutamos ahora nuestro programa veremos que nada ha cambiado, ambos canales siguen funcionando del mismo modo como ya hemos visto, esto es debido a que la línea 30 no altera el valor con que el registro 7 es inicializado por el ordenador. Tampoco sucederá nada si introducimos la línea 30 del siguiente modo:

```
30 SOUND 7, &B11100
```

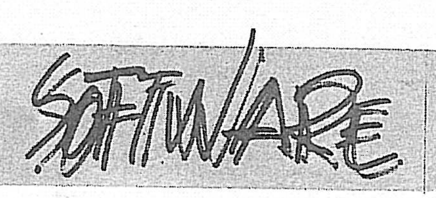
Ya que todavía el registro 7 mantiene activadas las salidas de tono de los dos canales (A y B) que hemos programado. En cambio carguemos ahora el mencionado registro con el siguiente valor:

```
30 SOUND 7, &B100100
```

Como podéis observar, ahora el sonido emitido por nuestro ordenador es bastante distinto, ya que hemos activado las salidas de ruido de nuestros dos canales y ambos están emitiendo tono y ruido al mismo tiempo. Desactivemos ahora las salidas de tono:

```
30 SOUND 7, &B100111
```

y comprobaremos que de nuevo el sonido generado porel PSG se ve modi-



ficado. Por último vamos a ver como el registro siete es capaz de silenciar totalmente la emisión de sonido de nuestro ordenador desactivando todas las salidas de tono y de ruido del PSG del siguiente modo:

30 SOUND 7, &B111111

Si activamos la salida de ruido de un canal y desactivamos el resto de salidas haciendo por ejemplo 10 SOUND 7, &B110111, podremos observar que el nivel de ruido no varia por más que modifiquemos los valores de los registros de **control de tono** (en nuestro ejemplo los registros 0 y 1). Podemos hacer la prueba introduciendo la siguiente linea: 30 SOUND 1,15: SOUND 0,200 y sustituyendo los valores de dichos registros por cualquier otro sin que el sonido por el PSG varie lo más mínimo.

Esto es debido a que los registros 0 y 1 del canal A o los registros 2 y 3 del B o 4 y 5 del canal C son **registros de control de tono** y no de control de ruido. Para controlar el nivel de ruido, el AY-3-8910 utiliza el registro 6 del cual sólo sus cinco bits menos significativos sirven para alterar dicho nivel de ruido. Los tres restantes bits de dicho registro no se utilizan. Así los valores entre los que podremos variar dicho nivel de ruido iran desde 00000 BINARIO = 0 decimal, hasta 11111 BINARIO = 31 decimal. Siempre que utilicemos este registro usaremos notación decimal teniendo siempre en cuenta no rebasar el valor 31 que es el máximo nivel de ruido que el PSG puede alcanzar. Veamos lo que sucedería si dieramos al registro 6 un valor mayor que 31, por ejemplo el valor 33, el número binario que corresponde al decimal 33 es 100001, como el registro 6 no utiliza mas que sus cinco bits menos significativos ignorará el uno situado más a la izquierda de dicho número binario y por lo tanto sólo tendrá en cuenta el valor 00001, binario = 1 decimal.

Si introducís estas líneas en el ordenador

10 SOUND 7, &B110111
20 SOUND 8,15, SOUND 6,1

y las ejecutáis y luego sustituís el valor 1 del registro 6 por 33 volviendolas a ejecutar, comprobaréis en la práctica lo expuesto anteriormente.

Desgraciadamente el PSG no posee más que un registro (el 6) para controlar el nivel de ruido de los tres canales. Por lo tanto si hacemos que dos o más canales emitan ruido simultáneamente, podremos conseguir que cada uno lo haga con un volumen diferente gracias a los registros 8,9 y 10 pero el nivel o grado de ruido será siempre idéntico para los tres y vendrá determinado por el valor que asignemos al mencionado registro 6.

Con lo explicado hasta aquí y un poco de práctica ya tenemos a nuestro alcance la posibilidad de enriquecer nuestros programas con una amplia gama de efectos sonoros. Como ejemplo aquí os ofrecemos una pequeña rutina que simula, cada vez que se pulsa la barra espaciadora, la caida de una bomba, introducid el siguiente listado, subid al máximo el volumen del televisor o monitor y ejecutad el programa, vereis que el efecto es bastante real.

Listado 1

```
10 ' Sondeo de barra espaciadora
20 '
30 STRIG(0)ON
40 ON STRIG GOSUB 1000
50 GOTO 50
60 '
970 '  Desactivacion del sondeo
980 'y silbido de caida de bomba
990 '
1000 STRIG(0) OFF
1010 SOUND 7,&B111110
1020 FOR I=20 TO 80
1030 SOUND 8,I/6:SOUND 0,I
1040 FOR J=1 TO 25:NEXT
1050 NEXT
1060 '
1070 '    Efecto de explosión
1080 '
1090 SOUND 7,&B101101
1100 SOUND 6,31
1110 SOUND 9,15:SOUND 3,15
1120 FOR H=600 TO 200 STEP-1
1130 SOUND 9,H/40
1140 NEXT
1150 '
1160 '  Desactivación del PSG y
1170 'reinicialización de volumen
1180 '
1190 SOUND 7,&B111111:SOUND 8,0
1200 '
1210 ' Reactivación del sondeo
1220 '  y regreso al programa
1230 '
1240 STRIG(0) ON:RETURN
```

Todavía nos restan por explicar tres registros del PSG (11,12 y 13), que se encargan del control de envolventes.

Habíamos visto como los registros 8,9 y 10 determinaban el volumen de cada canal para valores comprendidos entre 0 y 15 (1111 bin.). Esto es, dichos registros utilizan sus cuatro Bits menos significativos para controlar la amplitud o volumen de cada canal. Así si quisieramos por ejemplo hacer que el canal emitiese un ruido cuyo volumen subiera lentamente desde 0 hasta quince y volviese a bajar y así repetidas veces con lo que hemos visto hasta ahora, tendríamos que hacer un listado más o menos como el siguiente:

Listado dos

```
10 SOUND 7,&B110111
20 I=0
30 I=I+1
40 SOUND 8,I:SOUND 6,10
50 FOR J=1 TO 125:NEXT
60 IF I<16 THEN 30 ELSE GOSUB 70: GOTO 20
70 FOR J=1 TO 300:NEXT:RETURN
```

Como podeis ver este programa, además de ser bastante largo mantiene ocupado a la CPU de nuestro ordenador en incrementar y decrementar los valores del registro 8 sin que ésta pueda realizar al mismo tiempo otras tareas.

Sin embargo utilizando los registros de control de envolventes podemos reducir el programa anterior a las siguientes lineas:

Listado 3

```
10 SOUND 7,&B110111
40 SOUND 12,40:SOUND 13,14
50 SOUND 6,10:SOUND 8,16
```

Ejecutad este pequeño listado y comprobaréis que funciona prácticamente igual que el anterior, sólo que este último además de ser bastante más corto deja libre a la CPU para que ésta pueda realizar otros trabajos mientras el PSG funciona por su cuenta sin necesidad de que le introduzcamos nuevos valores.

Si os fijais en la linea 20 de éste pequeño programa observaréis que hemos dado al registro 8 valor 16 (10000 bin.), cuando anteriormente habíamos dicho que los registros de control de amplitud (8, 9 y 10) sólo utilizaban sus cuatro bits menos significativos para regular el volumen de cada canal y por lo tanto el máximo valor que podían alcanzar era 1111 BIN = 15 decimal. Pues bien, al dar a un registro de control de amplitud valor 16 (10000 bin.), lo que estamos haciendo es **activar** el quinto bit de dicho registro con lo que hacemos que el control de amplitud de su canal pase a estar a cargo de la envolvente u onda portadora que elijamos según el valor que demos al registro 13.

Este registro utiliza sus cuatro bits menos significativos para generar distintos tipos de envolventes, así pues

sería lógico pensar que el PSG posee 16 hondas portadoras diferentes cuando en realidad y de un modo anómalo sólo tiene ocho.

En el ejemplo anterior vimos como la envolvente numero 14 (SOUND 13, 14) hacia que el volumen del canal dependiente de ella partiera desde su valor mínimo hasta el máximo (0-15) y volviese a descender al mismo ritmo

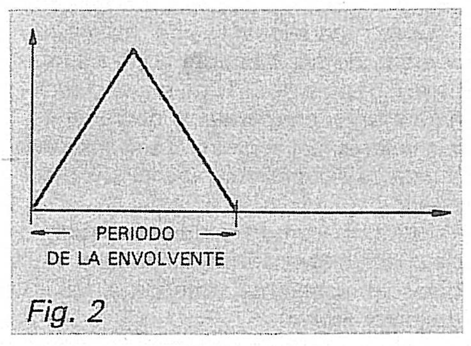


Fig. 2

desde el valor máximo de volumen al mínimo (15-0) y así sucesivamente. Es decir, que en un tiempo determinado la onda portadora parte de 0, alcanza su valor máximo (15) y vuelve a descender (a 0). Este **tiempo** se denomina **periodo de la envolvente** (fig. 2) y será controlado por los registros 11 y 12 como veremos a continuación. Pero primero vamos a conocer de modo gráfico cuales son los distintos tipos desenvolventes que podemos seleccionar desde el registro 13.

Puesto que ya habíamos explicado cómo se comporta la envolvente número 14 y atendiendo a su gráfica podréis ver facilmente cómo funcionan el resto de ondas portadoras.

Sólo una observación más; no es posible fijar una envolvente distinta para cada canal, así pues todos los canales en cuyos registros de amplitud se active el bit numero 5 (es decir, tomen valor 16) pasarán a depender de una misma envolvente.

Los registros 11 y 12 actuan conjuntamente en el control del periodo de la envolvente, ambos registros pueden utilizar para sus 8 bits, es decir, ambos pueden tomar valores entre 0 y 255 pero para el PSG los valores que tome el registro 12 son mucho más significativos que los que tome el registro 11. Es decir, si el registro 12 toma valor 255 el periodo de la envolvente será mucho mayor que si es el registro 11 el que adquiere dicho valor.

Esperamos que este artículo os haya servido para desvelar el funcionamiento del generador programable de sonidos de vuestro ordenador. De todos modos para que practiquéis un poco más, en la sección de trucos os ofrecemos algunos ejemplos que seguramente sabréis aprovechar.

L.A.S.A.

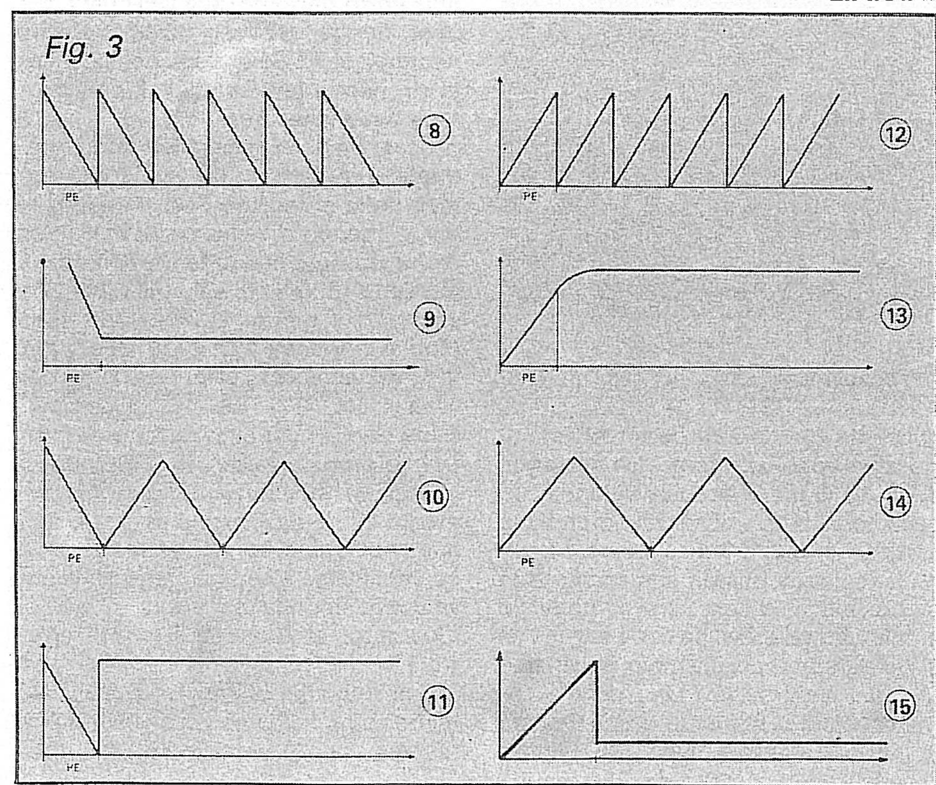


Fig. 3

GENERADOR DE SONIDOS

Como complemento a los dos artículos referentes al sonido hemos incluido éste programa. Se trata de un editor de sonidos con el que podrás crear efectos sonoros de gran variedad moviendo el cursor por la pantalla.

En pantalla se muestran los 14 registros que podemos modificar con su valor en binario y un pequeño comentario de la función que realizan. Además también nos aparecerá el cursor. Este podemos desplazarlo con flechas de dirección. Pulsando la barra modificamos el bit sobre el que se encuentra. Se hará 1 si es 0 y 0 si es 1.

También se incluyen varias funciones más accesibles desde teclado con las siguientes teclas:

= (+). Suma uno al dato y nos muestra como resultado un número en decimal.

—. Resta uno al dato binario y muestra el resultado en decimal como en el caso anterior.

ENTER (RETURN). Carga el registro indicado por el cursor con el valor que se le haya dado.

C. Borra todos los registros poniéndolos a cero.

D. Muestra en la línea de mensajes una sentencia data con los datos necesarios para producir los sonidos que hayas creado.

CTROL + STOP. Se abandona el programa.




```
1000 ' ************************
       *                      *
       * GENERADOR DE SONIDOS *
       *                      *
       ************************
1010 '
1020 '
1030 '
1040 '
1050 KEYOFF
1060 BEEP
1070 SCREEN 3,,0
1080 COLOR 1,15,15
1090 CLS
1100 A=RND(-TIME)
1110 OPEN "GRP:" AS#1
1120 PRESET (0,0)
1130 PLAY "V15T200O4BBBAGFEDCO3BAGFED
CO2BAGFEDCCC","V15T200O6BBBAGFEDCO5BA
GFEDCO4BAGFEDCCC"
1140 FOR A=1 TO 60
1150 COLOR 1+RND(1)*13
1160 PRINT#1," SOUND»"
1170 NEXT A
1180 CLOSE
1190 FOR A=1 TO 1000
1200 NEXT A
1210 ON STOP GOSUB 1810
1220 STOP ON
1230 SCREEN 0,,0
1240 COLOR 1,15
1250 LOCATE ,,0
1260 PRINT"***************************
************ GENERADOR DE SONIDOS **
CREASOUND **************************
*********** "
1270 PRINT"00: 00000000 -Tono A: Byte
 Inferior  01: 00000000         Byte
 Superior"
1280 PRINT"02: 00000000 -Tono B: Byte
 Inferior  03: 00000000         Byte
 Superior"
1290 PRINT"04: 00000000 -Tono C: Byte
 Inferior  05: 00000000         Byte
 Superior"
1300 PRINT"06: 00000000 -Generador Ru
ido (0-31) 07: 00000000 -Habilita Voz
 y Ruido   08: 00000000 -Amplitud voz
 A         09: 00000000 -Amplitud voz
 B         10: 00000000 -Amplitud voz
 C         11: 00000000 -Periodo env.
 byte inf."
1310 PRINT"12: 00000000
 byte sup. 13: 00000000 -Forma de la
envolvente  "
1320 PRINT"***************************
***********"
1330 DIM S$(13)
1340 DEF FN B$(N)=RIGHT$("00000000"+B
IN$(N),8)
1350 FOR A=0 TO 13
1360 SOUND A,0
1370 S$(A)="00000000"
1380 LOCATE 4,A+4
1390 PRINTS$(A);
1400 NEXT A
1410 X=0:Y=0
1420 D=STICK(0)
1430 IFD=1ORD=2ORD=8THENX=X-1:IFX<0TH
ENX=0
1440 IFD=4ORD=5ORD=6THENX=X+1:IFX>13T
HENX=13
1450 IFD=2ORD=3ORD=4THENY=Y+1:IFY>7TH
ENY=7
1460 IFD=6ORD=7ORD=8THENY=Y-1:IFY<0TH
ENY=0
1470 LOCATE Y+4,X+4,1
1480 IFSTRIG(0)=-1THENGOSUB 1560
1490 A$=INKEY$
1500 IF A$="=" THEN GOSUB 1610
1510 IF A$="-" THEN GOSUB 1640
1520 IF A$=CHR$(13) THEN GOSUB 1590
1530 IF A$="C" THEN BEEP:GOTO 1350
1540 IF INKEY$="D" THENGOSUB 1730
1550 GOTO 1420
1560 A$=MID$(S$(X),Y+1,1)
1570 IFA$="1"THENMID$(S$(X),Y+1,1)="0
":LOCATE Y+4,X+4,0:PRINT"0"
1580 IFA$="0"THENMID$(S$(X),Y+1,1)="1
":LOCATE Y+4,X+4,0:PRINT"1"
1590 SOUND X,VAL("&B"+S$(X))
1600 RETURN
1610 N=VAL("&B"+S$(X)):N=N+1:IFN>255
THEN N=255
1620 S$(X)=FNB$(N):LOCATE4,X+4,0:PRIN
TS$(X):LOCATE 9,20:PRINT"DATO EN DECI
MAL:";N;"    "
1630 GOTO 1590
1640 N=VAL("&B"+S$(X)):N=N-1:IFN<0 TH
EN N=0
1650 GOTO 1620
1660 LOCATE ,,0
1670 FOR A=20 TO 22
1680 LOCATE 0,A
1690 PRINT"
      ";
1700 NEXT A
1710 LOCATE 0,21
1720 RETURN
1730 GOSUB 1660
1740 PRINT"DATA ";
1750 FOR A=0 TO 13
1760 B=VAL("&B"+S$(A))
1770 PRINT RIGHT$("0"+HEX$(B),2);
1780 IF A<13 THEN PRINT",";
1790 NEXT A
1800 RETURN
1810 BEEP
1820 LOCATE ,,0
1830 STOP OFF
1840 SCREEN 0
1850 RETURN 1860
1860 PRINT"FIN"
1870 PRINT:END
```

FICHERO DE PANTALLAS

Os ofrecemos una rutina capaz de almacenar y luego imprimir más de 40 pantallas, empleando la memoria paginada.

Realmente con una rutina de una longitud despreciable, 127 octetos de memoria, mas un buffer de 768 bytes, donde almacenar la pantalla temporalmente, podemos por medio de la paginación de la memoria, almacenar hasta 42 pantallas en SCREEN 1 (modo de texto de 32 columnas), en los 32 kbytes de los que disponemos al librarnos de la ROM.

Una pantalla en SCREEN 1 ocupa en la memoria de video una longitud de 32 columnas x 24 filas = 768 bytes, por esto, en los 32768 bytes que obtenemos al paginar la memoria (direcciones de memoria 0-32767), podemos almacenar, la friolera cifra de 32768/768 = 42.6666667 pantallas, mejor dicho, 42 pantallas, ya que la última nos "macharía" la zona del programa basic.

Si deseais ver la rutina en acción tendréis que teclear el listado ensamblador aquellos que dispongais de él, y el cargador BASIC, los que no lo tengais.

Antes de problarla te recomiendo grabar todo en cinta, por si al teclear has cometido un error. Si usas ensamblador graba el fichero fuente y luego la rutina con:
BSAVE "CAS:FICHPAN",&HD2FO,&HD36E

Los que copien el cargador BASIC, la comprobación de CHECKSUM (suma de todos los datos introducidos) indicará si se ha cometido algún error. De ser así, revisa las líneas DATAS. Si el programa está libre de errores, continuará grabando la rutina en cinta, pulsando la tecla SPACE.

En este punto, ya estamos preparados para introducir las dos demostraciones:

- La primera de ellas, DEMOSTRACION 1, imprime en BASIC pantallas de la 0 a la 40, estas las almecena en memoria y luego las saca una tras otra.
- La DEMOSTRACION 2, simula un efecto de animación, sacando rápidamente 21 pantallas, una tras otra.

Lo primero que hace el programa es dibujar círculos concéntricos cada vez mayores, y cada círculo es almacenado como si de un fotograma se tratase.

A continuación, sacando las pantallas rápidamente, vemos como un círculo parece agrandarse en la pantalla y luego reducirse. Si deseas que sólo se agrande, cambia la línea 330 por: 330 GOTO 290

A continuación vamos a detallar las características de la rutina y la forma de usarla:

— Comienza la dirección de memoria 54.000 y finaliza en la 54.126.

— El buffer que se usa para almacenar temporalmente la pantalla, ocupa las posiciones 54127-54894.

— El número de la pantalla que vamos a imprimir o guardar, hay que POKEarlo, en la dirección 54012 (&HD2FC). Ejemplo:

LISTADO ENSAMBLADOR

PAGE 1

```
                        ; MANEJO DE PANTALLAS
                        ; MEDIANTE
                        ; LA MEMORIA PAGINADA
                        ;
                        ; ANGEL GARCIA DELGADO
                        ; MUNDO MSX -1987-
                        ;
                                    ORG   54000
                                    LOAD 54000
                        ;
D2F0 C32AD3                         JP    METE          ; PANTALLA > PAGINADA
D2F3 C33DD3                         JP    SACA          ; PAGINADA > PANTALLA
D2F6 C32DD3                         JP    METE2         ; BUFFER > PAGINADA
D2F9 C344D3                         JP    SACA2         ; PAGINADA > BUFFER
                        ;
D2FC 00                 PANTA:      DEFB 0              ; NUMERO PANTALLA
D2FD 00                 DATO:       DEFB 0
                        ;
D2FE 3AFCD2             OPERA:      LD    A,(PANTA)
D301 57                             LD    D,A
D302 1E00                           LD    E,0
D304 210000                         LD    HL,0
D307 0603                           LD    B,3
D309 19                 SUMA:       ADD   HL,DE
D30A 10FD                           DJNZ SUMA           ; HL=DIR. MEMORIA PAG.
D30C 116FD3                         LD    DE,BUFFER     ; DE=DIR. BUFFER
D30F C9                             RET
                        ;
D310 210018             VIDRAM:     LD    HL,6144
D313 116FD3                         LD    DE,BUFFER
D316 010003                         LD    BC,768
D319 CD5900                         CALL 89             ; PASA VRAM A RAM
D31C C9                             RET
                        ;
D31D 216FD3             RAMVID:     LD    HL,BUFFER
D320 110018                         LD    DE,6144
D323 010003                         LD    BC,768
D326 CD5C00                         CALL 92             ; PASA RAM A VRAM
D329 C9                             RET
                        ;
D32A CD10D3             METE:       CALL VIDRAM
                        ;
D32D CDFED2             METE2:      CALL OPERA
D330 CD53D3                         CALL PAGON
D333 EB                             EX    DE,HL
D334 010003                         LD    BC,768
D337 EDB0                           LDIR
D339 CD68D3                         CALL PAGOFF
D33C C9                             RET
                        ;
D33D CD44D3             SACA:       CALL SACA2
D340 CD1DD3                         CALL RAMVID
D343 C9                             RET
                        ;
D344 CDFED2             SACA2:      CALL OPERA
D347 CD53D3                         CALL PAGON
```

```
D34A 010003                 LD     BC,768
D34D EDB0                   LDIR
D34F CD68D3                 CALL   PAGOFF
D352 C9                     RET

PAGE    2

                  ;
D353 F3           PAGON:    DI                          ; PAGINA MEMORIA
D354 0EA8                   LD     C,168
D356 ED78                   IN     A,(C)
D358 32FDD2                 LD     (DATO),A
D35B 47                     LD     B,A
D35C CB38                   SRL    B
D35E CB38                   SRL    B
D360 CB38                   SRL    B
D362 CB38                   SRL    B
D364 B0                     OR     B
D365 D3A8                   OUT    (168),A
D367 C9                     RET
                  ;
D368 3AFDD2       PAGOFF:   LD     A,(DATO)             ; DESPAGINA
D36B D3A8                   OUT    (168),A
D36D FB                     EI
D36E C9                     RET
                  ;
                  BUFFER:   DEFS 768
                  ;
                            END
```

LISTADO DE ETIQUETAS

```
PAGE    1

BUFFER      D36F DATO      D2FD METE      D32A METE2     D32D
OPERA       D2FE PANTA     D2FC PAGON     D353 PAGOFF    D368
RAMVID      D31D SUMA      D309 SACA      D33D SACA2     D344
VIDRAM      D310
```

Si hacemos POKE &HD2FC,10 todas las operaciones que se realicen serán referentes a la pantalla 10 de la memoria paginada.

— La rutina tiene cuatro entradas posibles, cada una de ellas realiza una función diferente:

- Llamando a la dirección 54000 (&HD2FC), el contenido actual de la pantalla de vídeo es guardado en la memoria. El número de la pantalla donde se almacena es el indicado en la posición de memoria 54012.
- Para sacar la pantalla deseada de la memoria a nuestra pantalla, tendremos que llamar a la dirección 54003 (&HD2F3). En la posición 54012 debe estar el número de pantalla a sacar.

Las dos entradas que vienen a continuación han sido pensadas para que sea posible la grabación y la posterior carga de pantallas en la memoria paginada:

- Llamando a la posición 54006 (&HD2F6), el contenido del BUFFER es almacenado en la memoria. (Operación de carga, LOAD).
- Para pasar una pantalla de la memoria al BUFFER, hay que llamar a la dirección 54009 (&HD2F9). Para grabar la pantalla en cassette tendrás que grabar el contenido del buffer.

Si no comprendes claramente como se cargan y se graban las pantallas de la memoria paginada, estudia los listados de demostracion 3 y 4. La demostración 3 realizaría el proceso de carga, LOAD, y la demostración 4 el de grabación, SAVE.

El número de pantallas que se desean cargar o grabar está especificado entre ambos ejemplos, por la variable P del bucle FORNEXT.

Hasta aquí creemos haber proporcionado la información necesaria para el control de la rutina, no obstante, aquellos que deseen conocer su funcionamiento pueden continuar leyendo.

Como seguramente sabrás, el Z-80 puede manejar un área de memoria de hasta 64 k. Pero con la ayuda del "PPI 8255", estos 64 k pueden ser seleccionados de un banco de memoria mayor. Este banco está formado por cuatro "ranuras" que pueden contener, cada una, hasta 64 k. Se selecciona la memoria en bloques de 16 k llamados páginas, de forma que el Z-80 puede direccionar 4 de estas páginas.

Para realizar esta selección de páginas, hemos de enviar por el puerto A del PPI (&HA8) un byte, donde cada pareja de bits indica de qué ranura se ha de tomar cada una de estas cuatro páginas. Los 2 bits menos significativos para la página 0 (0-16383), los dos siguientes para la página uno (16384-32767), y así, sucesivamente.

Hay que tener en cuenta que una página únicamente puede estar ante el procesador, en la misma situación que ocupa en su ranura.

Los ordenadores MSX disponen de 4 ranuras, una de ellas es ROM, otra es RAM, y las otras dos se usan para incorporar cartuchos o como bus de expansión. La situación de estas ranuras, cambia de unas firmas a otras, por ello a la hora de paginar, es necesario seguir un proceso que nos asegure la total compatibilidad.

La rutina de paginación que hemos usado, la puedes ver en las líneas 62-73 del listado ensamblador. Su misión será sustituir los 32 k de ROM por los 32 k de RAM de la otra ranura.

Lo primero que hacemos es desactivar las interrupciones, luego leemos el valor actual del puerto A del PPI, y almacenamos su contenido en una posición de memoria etiquetada como DATO, para recuperarlo al "despaginar" la memoria.

En el dato leido, los cuatro bits superiores nos dicen en qué ranura están las páginas de RAM, por ello lo que hacemos es copiarlos en la mitad inferior

```
10 ' CARGADOR BASIC
20 '
30 ' FICHERO PANTALLAS
40 '
50 CLEAR 200,53999!
60 RESTORE 9000
70 S=0
80 FOR A=&HD2F0 TO &HD36E
90 READ B
100 S=S+B
110 POKE A,B
120 NEXT
130 IF S<>16159 THEN PRINT"ERROR EN DATA
S":END
140 CLS
150 PRINT"PULSA SPACE PARA GRABARLO"
160 IF STRIG(0)<>-1 THEN 160
170 BSAVE "CAS:FICH-PAN",&HD2F0,&HD36E
180 END

 9000 DATA 195,42,211,195,61,211,195,45,
211,195,68,211,14,80,58,252
 9010 DATA 210,87,30,0,33,0,0,6,3,25,16,
253,17,111,211,201
 9020 DATA 33,0,24,17,111,211,1,0,3,205,
89,0,201,33,111,211
 9030 DATA 17,0,24,1,0,3,205,92,0,201,20
5,16,211,205,254,210
 9040 DATA 205,83,211,235,1,0,3,237,176,
205,104,211,201,205,68,211
 9050 DATA 205,29,211,201,205,254,210,20
5,83,211,1,0,3,237,176,205
 9060 DATA 104,211,201,243,14,168,237,12
0,50,253,210,71,203,56,203,56
 9070 DATA 203,56,203,56,176,211,168,201
,58,253,210,211,168,251,201
```

```
10 ' DEMOSTRACION 1
20 '
30 DEFUSR1=54000!
40 DEFUSR2=54003!
50 DEFUSR3=54006!
60 DEFUSR4=54009!
70 '
80 FOR A=0 TO 40
90 CLS
100 FOR B=0 TO 10
110 PRINT "PANTALLA",A
120 NEXT
130 POKE &HD2FC,A
140 B=USR1(0)
150 NEXT
160 '
170 FOR A=1 TO 30
180 BEEP
190 NEXT
200 '
210 FOR A=0 TO 40
220 POKE &HD2FC,A
230 B=USR2(0)
240 FOR B=0 TO 50
250 NEXT
260 NEXT
```

del byte, para que las cuatro páginas sean tomadas de la ranura de RAM.

Este dato lo enviamos de nuevo por el canal A del PPI, quedando de esta forma paginada la memoria.

La rutina con la que "despaginamos" (líneas 75-78), es muy sencilla, se reduce a recuperar el valor contenido en DATO y sacarlo por el canal A del PPI para recuperar la situación original. Luego se activán las interrupciones.

Es muy importante mantener las interrupciones desactivadas mientras la memoria esté paginada, ya que al no disponer de la ROM, tendríamos que controlar el salto que se realiza al producirse la interrupción.

El resto de las subrutinas son de gran sencillez:

OPERA, calcula a partir del número de pantalla contenido en la etiqueta PANTA, la dirección en la memoria paginada. Este valor es devuelto en el registro HL, y en DE se carga la dirección de inicio del BUFFER.

VIDRAM, se encarga de pasar el contenido de la memoria de video a nuestro BUFFER, en la RAM.

RAMVID, realiza la operación inversa de VIDRAM, pasa el contenido del BUFFER a la pantalla de video.

METE, pasa la pantalla de video a la memoria paginada. Para hacerlo, llama a VIDRAM, luego a OPERA, e intercambia los valores de HL y DE, se activa la paginación, se transfiere el bloque de datos de la pantalla, se despagina y se regresa al BASIC.

METE2, es igual que METE pero no realiza la primera llamada a VIDRAM, por lo que tanto lo que hace es que pasa el contenido del BUFFER a la memoria paginada.

SACA, llama a SACA2, luego a RAMVID y regresa. Su función es pasar el contenido de la memoria paginada a la pantalla de VIDEO.

SACA2, pasa los datos de la pantalla en la memoria paginada, al BUFFER. Para hacerlo, llama a OPERA y PAGON para paginar la memoria y obtener las direcciones que nos van a permitir pasar el bloque de datos de 768 bytes mediante un sencillo LDIR. Hecho esto, se despagina y se retorna.

Esperamos que tras esta explicación os sea más fácil comprender como se pagina la memoria. Verdaderamente no es un tema complicado, pero hay que considerar tres puntos: la compatibilidad, la ausencia de la ROM y la acción de las interrupciones.

Posibles aplicaciones de la rutina FICHERO DE PANTALL serían: La creación de juegos con muchas pantallas, almacenamiento de texto en programas de tratamiento de texto, efectos de animación y un sin fín de posibilidades abiertas a tu propio ingenio.

```
10 ' DEMOSTRACION 2
20 '
30 DEFUSR1=54000!
40 DEFUSR2=54003!
50 DEFUSR3=54006!
60 DEFUSR4=54009!
70 '
80 PI=3.14159
90 FOR A=0 TO 20
100 SCREEN 1
110 PRINT "PA:";A
120 FOR B=0 TO 2*PI STEP PI/(A+1)
130 X=15+A*COS(B)
140 Y=11+A*SIN(B)
150 IF X<0 THEN 200
160 IF Y<0 THEN 200
170 IF X>31 THEN 200
180 IF Y>23 THEN 200
190 LOCATE X,Y:PRINT"*";
200 NEXT
210 POKE &HD2FC,A
220 B=USR1(0)
230 NEXT
240 '
250 FOR A=1 TO 30
260 BEEP
270 NEXT
280 '
290 FOR A=0 TO 20
300 POKE &HD2FC,A
310 B=USR2(0)
320 NEXT
330 BEEP
340 FOR A=20 TO 0 STEP -1
350 POKE &HD2FC,A
360 B=USR2(0)
370 NEXT
380 BEEP
390 GOTO 290
```

```
10 ' DEMOSTRACION 3
20 '
30 '  CARGA (LOAD)
40 '
50 DEFUSR1=54000!
60 DEFUSR2=54003!
70 DEFUSR3=54006!
80 DEFUSR4=54009!
90 '
100 BUFFER=&HD2F6
110
120 FOR P=0 TO 5
130 '
140 N$="CAS:"+STR$(P)
150 BLOAD N$
160 '
170 POKE 54012!,P
180 A=USR3(0)
190 '
200 NEXT
```

```
10 ' DEMOSTRACION 4
20 '
30 ' GRABACION (SAVE)
40 '
50 DEFUSR1=54000!
60 DEFUSR2=54003!
70 DEFUSR3=54006!
80 DEFUSR4=54009!
90 '
100 BUFFER=&HD2F6
110 '
120 FOR P=0 TO 5
130 '
170 POKE 54012!,P
180 A=USR4(0)
190 '
191 N$="CAS:"+STR$(P)
192 BSAVE N$,BUFFER,BUFFER+768
193 '
200 NEXT
```

CUBILETES

Como muchas veces habrás oido, "la mano es más rápida que la vista". Si deseas comprobarlo por tí mismo, aquí tienes éste programa. Existen tres cubiletes y bajo uno de ellos se esconde una pequeña bola que deberás encontrar. Al comienzo del juego el ordenador te mostrará donde se encuentra la bola y, a continuación, la cambiará de lugar.

Cubiletes es un juego sencillo donde se han empleado SPRITES de 16 x 16 ampliados (32 x 32) para darle una mayor vistosidad (a costa de perder en la resolución de las figuras).

1000-1090. Lo primero que hace el programa es preparar la pantalla. Para ello, desactiva los "letreros" de las teclas de función, apaga los sonidos que pudiese haber y se fija el modo de texto 1 con SPRITES ampliados de 16 x 16 y con una anchura en pantalla de 29 caracteres. Se dan los colores al papel, a la tinta y al borde, y se introducen los datos de los SPRITES, directamente en la VRAM para hacerlo nos servimos del comando VPOKE.

1100-1110. Dibujo de los marcadores que nos van a informar de los puntos conseguidos y de los fallos que tengamos. Se imprimen además los recuadros que contienen la numeración de los cubiletes.

1730-1150. Aquí pintamos por primera vez los cubiletes, ponemos las variables de los PUNTOS y la de los fallos a cero, y definimos en la variable P$, una función que nos permitirá transformar el número dado como parámetro a una cadena de cuatro dígitos.

1160-1220. Se imprime el título y suenan unos breves acordes. En la línea 1190 nos aseguramos de que el programa no continúe hasta no acabar la música. Se borra el título y se sitúa el SPRITE de la bola.

1230-1290. Se realizan varias llamadas a subrutinas, que son las encargadas de levantar y luego bajar los cubiletes, y se limpia el buffer del teclado.

1300-1440. Se imprime el mensaje de que es nuestro turno y espera la pulsación de las teclas "1", "2" ó "3", valor que se pasa a numérico y se guarda en la variable R, se realiza un sonido, se imprimen los marcadores y se levanta el cubilete elegido.

1480-1480. Se completa el bucle principal llamando a la subrutina de la 1910 en caso de descubrir la bola, y a la rutina de la 2000 en caso negativo.

A continuación, vienen las subrutinas donde se realizan las operaciones del programa, éstas son:

1490-1490. Inicializa el Sprite N.

1500-1540. Levanta el cubilete número N.

1550-1590. Baja el cubilete número <n.

1600-1760. Se fija la posición de la bola al azar y se dibuja.

1770-1800. Baja aquellos cubiletes que están levantados. Para hacerlo, consultamos con VPEEK el contenido de la VRAM.

1810-1840. Sube los cubiletes que estén bajados.

1850-1890. Imprime marcadores.

1900-1900. Ventana de texto de una línea.

1910-1990. Correcto. Sumar 5 puntos.

2000-2080. Incrementa el número de fallos. Si éstos son mayores que 4, salta al final.

2090-2210. Final. Mensaje de GAME OVER. Pulsando "S" se comienza y pulsando "N" el programa concluye.

2220-2280. Rutina que introduce los datos de los sprites en la memoria de video.

```
1000 ' *************
       *           *
       * CUBILETES *
       *           *
       *************
1010 '
1020 '
1030 '
1040 '
1050 KEYOFF
1060 BEEP
1070 SCREEN 1,3,0
1080 COLOR 1,2,2
1090 GOSUB 2220
1100 LOCATE 0,2:PRINT"    XWWWWWWWWY
 XWWWWWWWWY          V PUNTOS V    V FALLO
S V        V         V    V          V
 V  0000   V    V  0000   V          ZWWWWWW
WWC    ZWWWWWWWWC"
1110 LOCATE 0,17:PRINT"              XWY X
WY XWY                  V1V V2V V3V
                   ZWC ZWC ZWC"
1120 GOSUB 1730
1130 P=0
1140 F=0
1150 DEFFNP$(P)=RIGHT$("0000"+MID$(ST
R$(P),2,4),4)
```

```
1160 A$="--=< CUBILETES >=--"
1170 GOSUB 1900
1180 PLAY "V1504CDEGEEE","V1506CDEGEE
E"
1190 IF PLAY(0)=-1 THEN 1190
1200 A$=" "
1210 GOSUB 1900
1220 PUTSPRITE5,(115,90),4,1
1230 N=2
1240 GOSUB 1500
1250 GOSUB 1810
1260 GOSUB 1600
1270 FOR A=1 TO 50
1280 A$=INKEY$
1290 NEXT A
1300 A$="ADIVINA DONDE ESTA"
1310 GOSUB 1900
1320 A$=INKEY$
1330 IF A$<"1" OR A$>"3" THEN 1320
1340 R=VAL(A$)
1350 A$=" "
1360 GOSUB 1900
1370 BEEP
1380 SOUND 8,15
1390 FOR A=100 TO 250 STEP 7
1400 SOUND 0,A
1410 NEXT A
1420 SOUND 8,0
1430 N=R
1440 GOSUB 1500
1450 IF U=R THEN GOSUB 1910 ELSE GOSU
B 2000
1460 GOSUB 1770
1470 GOSUB 1850
1480 GOTO 1250
1490 PUTSPRITEN,(55+30*N,90),7,0:RETU
RN
1500 FORA=90 TO 65 STEP -1
1510 PUTSPRITEN,(55+30*N,A)
1520 BEEP
1530 NEXT A
1540 RETURN
1550 FOR A=65 TO 90
1560 PUTSPRITEN,(55+30*N,A)
1570 BEEP
1580 NEXT A
1590 RETURN
1600 U=RND(-TIME)
1610 U=1+INT(RND(1)*3)
1620 A$="!!! CAMBIO LA BOLA !!!"
1630 GOSUB 1900
1640 PUTSPRITE5,(55+30*U,90),4,1
1650 FOR A=1 TO 20
1660 SOUND 1,100+RND(1)*150
1670 SOUND 8,15
1680 FOR B=1 TO RND(1)*150
1690 NEXT B:SOUND 8,0
1700 FOR B=1TO 50
1710 NEXT B
1720 NEXT A
1730 FOR N=1TO3
1740 GOSUB 1490
1750 NEXTN
1760 RETURN
1770 FOR N=1TO3
1780 IF VPEEK(6912+4*N)=90 THEN GOSUB
 1500
1790 NEXTN
1800 RETURN
1810 FOR N=1TO3
1820 IF VPEEK(6912+4*N)=65THEN GOSUB
1550
1830 NEXT N
1840 RETURN
1850 LOCATE 6,5
1860 PRINTFNP$(P)
1870 LOCATE 19,5
1880 PRINTFNP$(F)
1890 RETURN
1900 LOCATE 0,8:PRINT"
                    ":LOCATE 0,8:PRINTTAB(1
5-LEN(A$)/2);A$:RETURN
1910 P=P+5
1920 GOSUB 1850
1930 SOUND 8,15
1940 FOR A=1 TO 10
1950 FOR B=0 TO 255 STEP A
1960 SOUND 0,B
1970 NEXT B
1980 NEXT A
1990 RETURN
2000 F=F+1
2010 GOSUB 1850
2020 SOUND 1,1:SOUND 8,15
2030 FOR A=150 TO 255 STEP .2
2040 SOUND 0,A
2050 NEXT A
2060 SOUND 8,0:SOUND 1,0
2070 IF F>4 THEN RETURN 2090
2080 RETURN
2090 GOSUB 1770
2100 GOSUB 1810
2110 A$="GAME OVER"
2120 GOSUB 1900
2130 FOR A=1 TO 2000
2140 NEXT A
2150 A$="OTRA PARTIDA (S/N)"
2160 GOSUB 1900
2170 POKE &HFCAB,255
2180 A$=INKEY$
2190 IF A$="S" THEN RUN
2200 IF A$="N" THEN SCREEN 0:END
2210 GOTO 2180
2220 DATA 7,C,19,13,37,27,27,33,3C,3F
,3F,3F,3F,3F,1E,7,E0,F0,F8,F8,FC,FC,F
4,CC,3C,FC,FC,FC,FC,FC,F8,E0,0,0,0,0,
0,0,0,0,0,1,2,3,3,1,0,0,0,0,0,0,0,0,0
,0,0,80,C0,C0,C0,80,0,0
2230 RESTORE 2220
2240 FOR A=14336 TO 14399
2250 READ B$
2260 VPOKE A,VAL("&H"+B$)
2270 NEXT A
2280 RETURN
```

HISTORIA

El super detective **Mike Bronco** de gran prestigio mundial ha sido contratado por el Alcalde de Chicago, con el fin de combatir y eliminar el crimen organizado que asola la ciudad en estos años veinte. Pero es un trabajo difícil a pesar de la categoría de Mike, pues ha de enfrentarse cuerpo a cuerpo contra los cinco grandes Jefes y sus respectivos secuaces que tienen dominada y aterrorizada toda la ciudad.

Estos son los cinco Capos contra los que debe luchar **Mike Bronco:**

— **Ruddy Bulldog:** jefe de los atracadores, el que en poco tiempo ha conseguido ser el único responsable de todo tipo de robos en la ciudad, desde bancos hasta pastelerías pasando por tiendas de frutos secos.

— **Jonhny Fandango:** Amo y señor de los contrabandistas de Chicago, controla los negocios de alcohol, cocaína y porcelana china.

— **Tony Spaguetty:** Adoptó el cargo de jefe de los extorsionadores de herencia, temido por toda la ciudad incluyendo al Alcalde así como el último borracho de la ciudad.

— **Franky Frondasio:** inspector de policía de profesión pero ejerce de todo lo contrario, permite y participa con sus adictos en todas las fechorías de los demás jefes, por lo que tratará de impedir que **Mike** termine su trabajo. Ya que ha dado orden de capturarlo vivo o muerto.

— **El padrino:** su historia es de sobra conocida.

¡Animo super detective! a recorrer toda la ciudad y elimina uno por uno a los héroes del hampa.

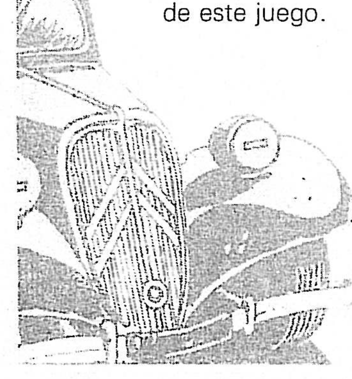
EL JUEGO

Has de buscar por las 92 pantallas de Chicago a los cinco grandes capos y eliminarlos. Antes de llegar a ellos, habrás de combatir con sus esbirros y al eliminarlos coger sus municiones, ya que las tuyas no son eternas.

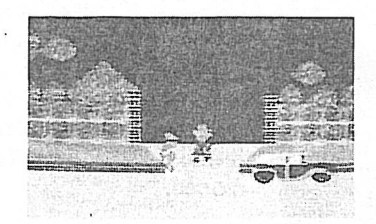


COSA NOSTRA es un juego emocionante que une a su gran calidad gráfica un nivel de dificultad elevado y gran número de diferentes pantallas (92); que constituye un reto para el jugador que siempre intentará llegar más lejos en su aventura.

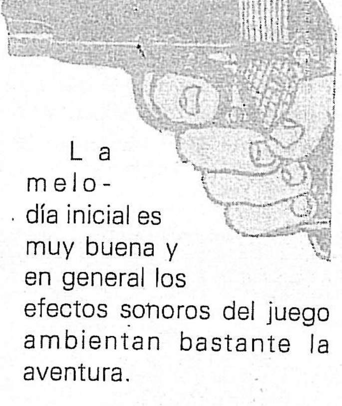
La respuesta de MIKE BRONCO a nuestras órdenes desde el teclado o joystick es muy rápida a pesar de ello la velocidad con que se mueven sus oponentes y la necesidad de tener que encararlos para poder dispararles hacen difícil el pasar con éxito las pantallas de este juego.



Opera-soft presenta "COSA NOSTRA" en dos formatos: disco de 3,5 pulgadas y cinta de cassette, por supuesto el tiempo de carga

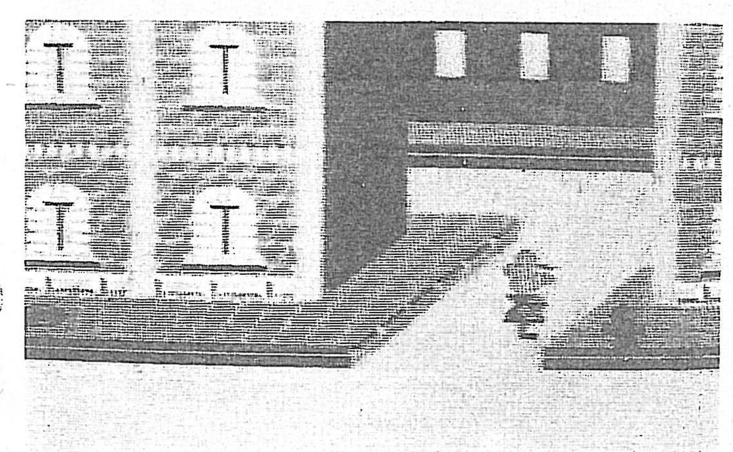


del disco es mucho más rápido y se efectúa "reseteando" el ordenador, mientras que el cassette utiliza BLOAD "CAS:"R.



La melodía inicial es muy buena y en general los efectos sonoros del juego ambientan bastante la aventura.

COSA NOSTRA es en definitiva un juego basado en una idea bastante original que combina perfectamente la acción con la vistosidad gráfica.



FUNKY PUNKY

FUNKY PUNKY es una aventura en la que su protagonista sale de su habitación para acudir a un examen. Para ello deberá recoger una serie de objetos que hay dispersos por las diferentes habitaciones de la casa y que le son necesarios para coger su moto y poder acudir al instituto, sorteando el tráfico, para realizar su examen.

Podemos dividir éste juego en cuatro partes: la primera transcurre dentro de la casa y en ella podemos contemplar una gran variedad de gráficos muy vistosos y elaborados, el nivel de dificultad de ésta fase no es muy elevado y basta ir encontrándo los diferentes objetos que aumentan nuestra energía y esquivando a los enemigos para pasar por el laberinto de pantallas que forman la casa. Una vez fuera de ella y con las llaves de la moto en la mano, se pasa a la segunda fase donde hay que ir adelantando los coches que vamos alcanzando, ésta etapa es bastante difícil de superar y en ella podemos apreciar un desplazamiento o scroll rápido y suave, y el movimiento de la moto también muy conseguido.

La tercera fase comienza al llegar al instituto; otro laberinto de pantallas similar al de la casa, donde hay que econtrar el aula del examen. Una vez en ella, y si llevamos los objetos pertinentes, comienza la cuarta y última fase: el examen, que consta de 10 difíciles preguntas a las que hay que responder afirmativa o negativamente.

En general los gráficos están muy elaborados aunque no hay una gran variedad de enemigos.

La pantalla del juego presenta un simpático dibujo acompañado de una melodía alegre y pegadiza.

La idea de la aventura es muy original aunque utilice el clásico recurso de las pantallas en forma de laberinto.

Como hemos dicho, la melodía inicial es muy buena y durante el resto del juego algunos sonidos van ambientando ligeramente las distintas situaciones.

¡Ahrrrg!, ¡qué dolor de cabeza!. No vuelvo a dar otra fiesta hasta que no pasen dos meses. ¡Tomá! si hoy tengo un examen de evaluación a las 10; y todavía aquí...

Realmente FUNKY PUNKY lo tiene difícil. Ha de ir al instituto para realizar el examen que tiene a las 10 de la mañana. Son las 8 y todavía tiene que recoger todas sus cosas revueltas entre los desperdicios de la fiesta que tuvo lugar en su apartamento la noche anterior.

Cuando las reuna, ha de coger su ciclomotor e ir al instituto atravesando las peligrosas calles de la ciudad, llenas de conductores medio dormidos que se dirigen a su trabajo.

Una vez allí, deberá encontrar el aula donde se va a realizar el examen y afrontar con toda resignación las diez preguntas que decidirán su aptitud para superar el curso.

Para realizar el examen será necesario que lleve su pluma y la calculadora científica.

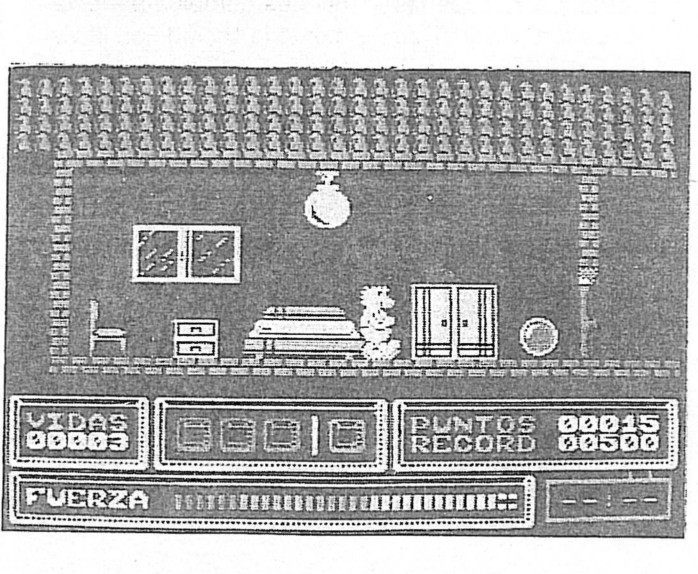
Por supuesto, no ha desayunado y para mantener sus energías a un alto nivel deberá tomar los DONUTS y beber los botellines que encuentre. Un cigarrillo después de la comida tampoco le sentará mal.

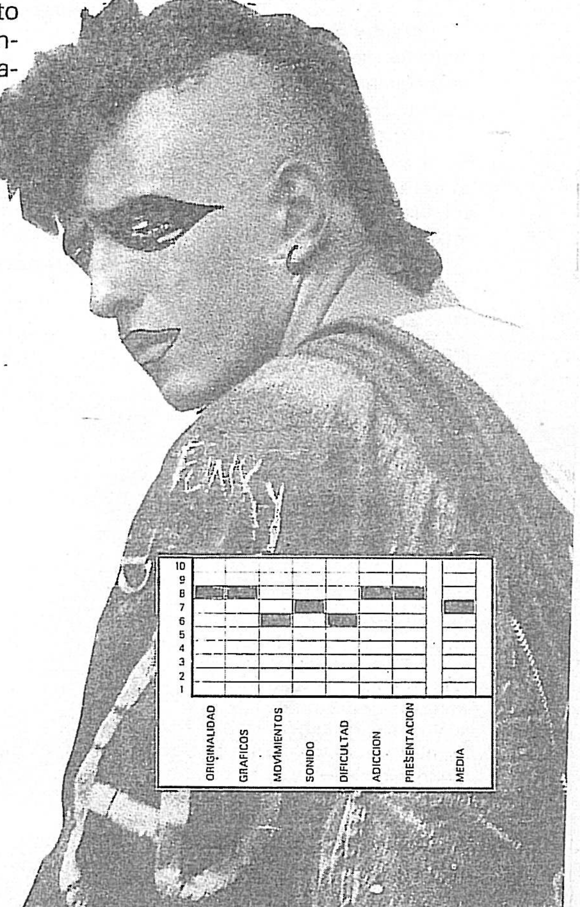
Los efectos de la resaca harán que vea bichos extraños que tendrá que evitar si desea continuar sus andanzas.

Para ver el tiempo que le resta hasta la hora del examen, deberá recoger el reloj. Y para poder abrir el candado de la moto tendrá que usar su llave.

¡Deprisa!, queda poco tiempo y el camino hasta el instituto es largo.



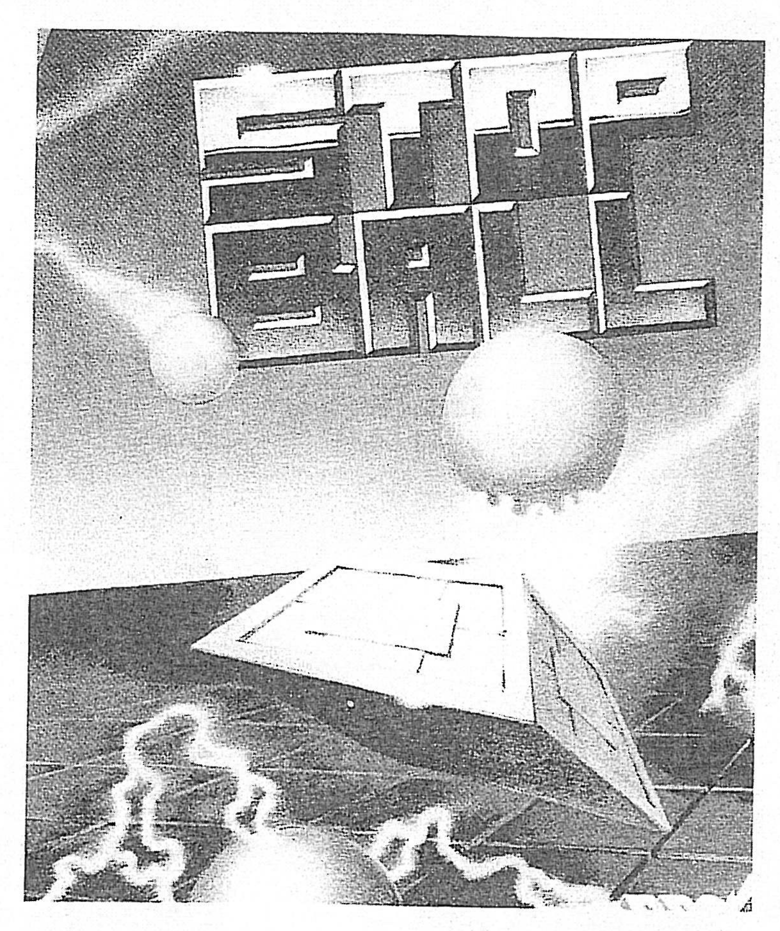


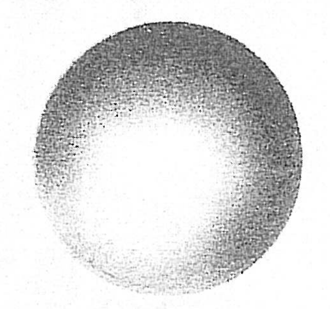


Como ya sabes, STOP BALL son dos juegos en uno, el primero es un juego en el que la velocidad resulta ser el factor principal; debes mantener en el aire todas las bolas, utilizando para ello tu paleta. En el segundo juego, has de tocar, antes de que se acabe el tiempo, todas las pirámides de base cuadrada que aparezcan en la pantalla, eludiendo al mismo tiempo las bolas que se arrastran por el suelo.

CONTADORES

Son tres y se encuentran en la parte derecha de la pantalla. El de más arriba es el reloj de cuenta atrás, el del centro el contador de energía y el inferior es el marcador. Tu energía disminuirá tanto si las bolas del primer juego tocan el suelo, como si eres tocado por las bolas del segundo. **Nota útil:** No debes tocar las cruces de las esquinas, ya que restan grandes cantidades de energía.



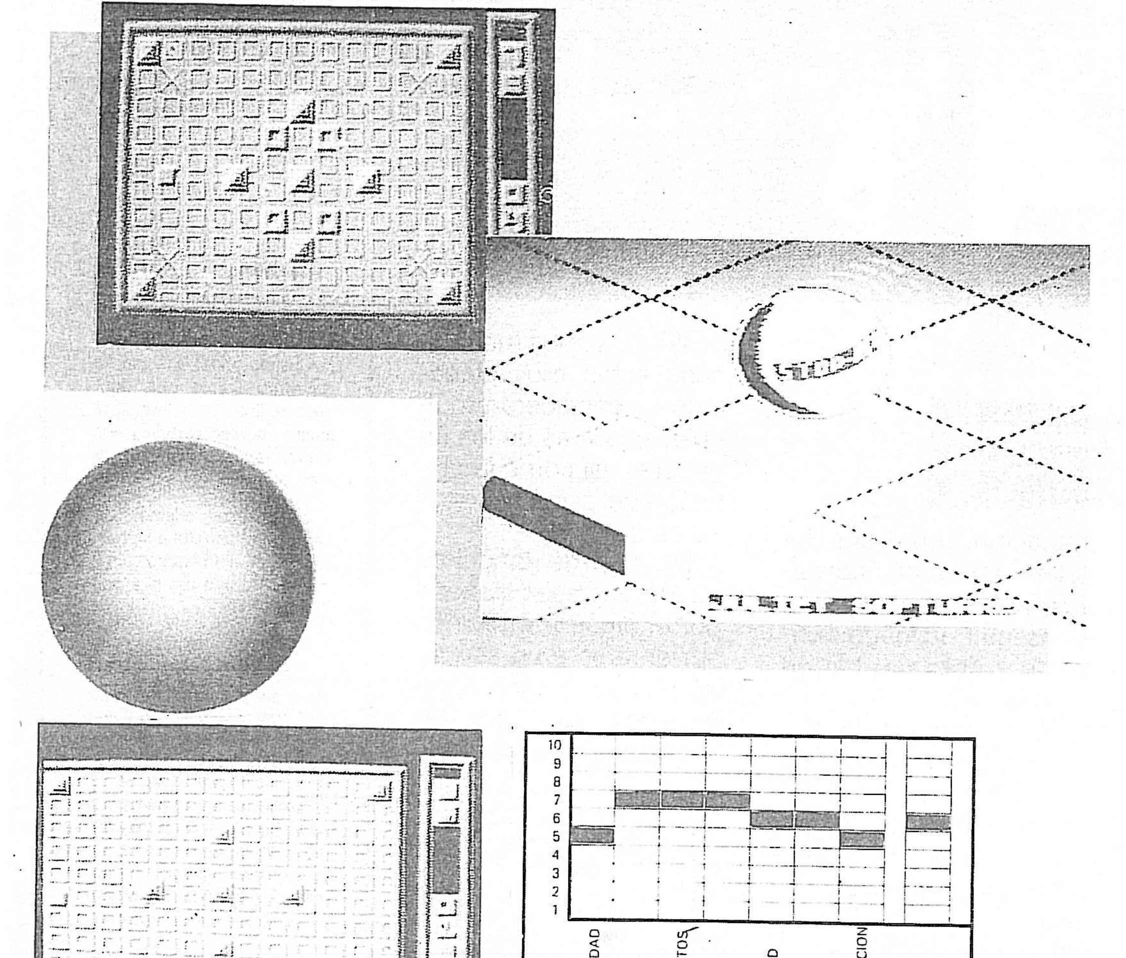


STOP BALL es un juego que presenta un efecto tridimensional muy conseguido: mediante la sombra que dejan las bolas en el suelo podemos conocer a qué altura se desplazan las mismas.

Los movimientos son suaves y rápidos y la respuesta de la paleta a las instrucciones del Joystik o teclado es perfecta.

Los gráficos, aunque en blanco y negro y no muy variados, consiguen bien el objetivo de dar una sensación de relieve tridimensional a los objetos que aparecen en pantalla.

A la derecha de la imagen se encuentran los marcadores en los que aparece nuestra puntuación en caracteres a tono con el juego. La dificultad no es muy elevada al principio pero va aumentando a lo largo de las diferentes fases del juego.



Una agradable melodía ameniza toda la partida y además pueden apreciarse algunos efectos sonoros cuando golpeamos las bolas.

La aventura no es muy original pero sí invita a tratar de superar todos los niveles de dificultad (32) de que se compone STOP-BALL.

FERNANDO MARTIN

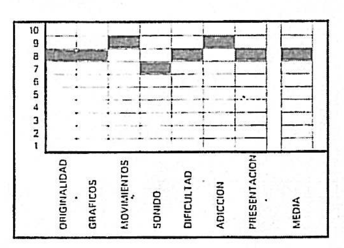
Basket Master

Los juegos de "simulación" entrañan a la hora de programarlos una dificultad superior a los demás dado que necesitan ajustarse lo más posible a la realidad que simulan, no basta con que el juego funcione bien y sea divertido sino que además debe parecer real, y esto implica un duro trabajo.

Fernando Martín Basket Master cumple perfectamente todo lo exigible a un juego de simulación, todos sus detalles están cuidadosamente estudiados para conseguir un juego excitante y divertido por su "realismo".

En la pantalla de presentación aparece una simpática caricatura del famoso jugador ligeramente animada y con una buena melodía. El resto de sonidos del juego están destinados a proporcionarle unos adecuados efectos ambientales. Los gráficos como ya hemos mencionado están muy elaborados y consideran múltiples posturas de los jugadores así como la agitación del público en cada canasta.

El nivel de dificultad puede ser seleccionado por el jugador pudiendo enfrentarse a un segundo jugador o bien al ordenador que hace las veces de Fernando Martín.



HISTORIA

Cuando salgas a la cancha te encontrarás nervioso, preocupado, con deseos de hacer un buen partido y con miedo de fallar los tiros de tres puntos.

Imagina que, además, tienes que enfrentarte a Fernando Martín, a un auténtico número uno, que no va a perdonarte.

Todos tus músculos se ponen en tensión, la adrenalina fluye por tus venas y tu cerebro debe averiguar cuáles son los puntos débiles de tu adversario.

El baloncesto es el deporte que mejor conjuga el desarrollo físico del ser humano, según un ordenador especializado en deportes del Istituto Tecnológico de Masachusets.

Fernando Martín Basket Master es una prueba clara de que también el baloncesto puede desarrollar la mente.



CAP INTERMITENTE

LISTADO ENSAMBLADOR

```
                        ; CAP INTERMITENTE
                        ; POR INTERRUPCIONES
                        ;
                        ; MUNDO MSX -1987-
                        ; ANGEL GARCIA DELGADO
                        ;
                        ;
                                    ORG   54000
                                    LOAD  54000
                        ;
 D2F0 C3FAD2                        JP    ACTIVA
 D2F3 C30FD3                        JP    DESACT
                        ;
 D2F6 64                VEL1:       DEFB  100
 D2F7 64                VEL2:       DEFB  100
 D2F8 00                POSI:       DEFB  0
 D2F9 01                CVEL:       DEFB  1
                        ;
 D2FA F3                ACTIVA:     DI
 D2FB 3EC3                          LD    A,0C3H
 D2FD 2117D3                        LD    HL,LUCE
 D300 DD219FFD                      LD    IX,0FD9FH
 D304 DD7700                        LD    (IX+0),A
 D307 DD7501                        LD    (IX+1),L
 D30A DD7402                        LD    (IX+2),H
 D30D FB                            EI
 D30E C9                            RET
                        ;
 D30F F3                DESACT:     DI
 D310 3EC9                          LD    A,201
 D312 329FFD                        LD    (0FD9FH),A
 D315 FB                            EI
 D316 C9                            RET
                        ;
 D317 F5                LUCE:       PUSH AF
```

Os ofrecemos una rutina que mediante interrupciones mantiene intermitente el piloto de la tecla CAP.

Para la creación de la rutina nos hemos servido de dos cualidades del MSX:

1. La existencia de vector de interrupción en la posición de memoria &HFD9F.
2. Una rutina de la ROM, que nos permita fijar el estado de encendido o apagado de la luz de CAP.

El restultado, es que con una rutina que sólo ocupa 77 bytes podemos hacer que la luz de CAP se mantenga intermitente mientras realizamos otras operaciones.

Además, nos es posible controlar el tiempo que debe estar encendida y el que tiene que transcurrir mientras se encuentre apagada.

Para observar como funciona tendrás que introducir el listado ensamblador o el cargador BASIC que se acompaña.

En ambos casos graba todo antes de probarla, pues si cometes algún error, lo más seguro es que el sistema "casque".

Los detalles de rutina son los siguientes:

— Comienza en &HD2F0 y acaba en la dirección &HD33C.

— Su longitud es de 77 bytes.

— No es relocalizable. Es decir, no puede ser cambiada de posición dentro de la memoria, a no ser que se ensamble cambiando la dirección de ORG.

— Para fijar el tiempo en unidades de 1/50 de segundo, hay que POKEar en la posición &HD2F6 un valor entre 0 y 255. Por ejemplo:

POKE %HD2F6, 50

Hará que la luz se mantenga encendida durante un segundo.

— Para determinar la duración de la pausa entre destello y destello, habrá que hacer lo mismo que en el caso anterior, pero en la dirección de memoria a &HD2F7. Por ejemplo:

POKE &HD2F7,25

hará que la pausa entre destello y destello sea de medio segundo.

— Para activar la interrupción, habrá que tener fijadas previamente las duraciones del destello y la pausa. Luego tendrás que hacer:

POKE &HD2F8,0

para asegurarse de que empieza apagada...

Y por último hacemos:

DEFUSR = &HD2F0: A = USR(0)

para activar las interrupciones. A partir de este momento, la lámpara permanecerá intermitente.

— Para desactivar la interrupción escribe:

DEFUSR1 = &HD2F3: A = USR1 (0)

y la lámpara dejará de lucir.

Aunque la luz se esté encendiendo y apagando constantemente, el estado de las mayúsculas no cambia, es decir, si estaba en minúsculas seguirá en minúsculas, aunque el piloto se encienda y viceversa.

Por supuesto, pulsando la tecla CAP se cambia el estado pero sin afectar al encendido del piloto.

Análisis de la Rutina

Como decíamos, disponemos de un "gancho" en FD9FH, que se utiliza para el control de las interrupciones generadas en el retorno de barrido de pantalla por el VDP. Esta interrupción controla la id exploración del teclado y varias funciones del sistema operativo. Por ello, es necesario, que en el retorno de cualquier rutina que creemos, se produzca la rutina de lla-

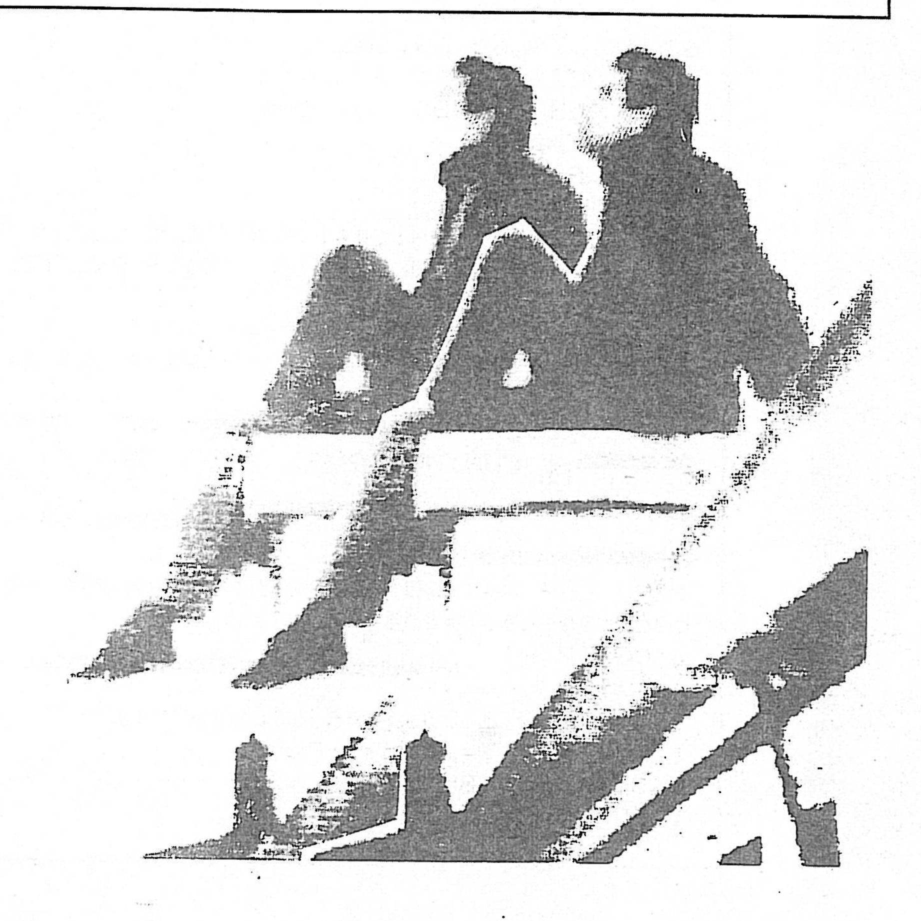
```
D318 3AF9D2           LD   A,(CVEL)
D31B 3D               DEC  A
D31C B7               OR   A
D31D 2019             JR   NZ,NO
D31F 3AF8D2           LD   A,(POSI)
D322 EE01             XOR  1
D324 32F8D2           LD   (POSI),A
D327 CD3201           CALL 0132H
D32A 3AF6D2           LD   A,(VEL1)
D32D 47               LD   B,A
D32E 3AF7D2           LD   A,(VEL2)
D331 32F6D2           LD   (VEL1),A
D334 78               LD   A,B
D335 32F7D2           LD   (VEL2),Á
D338 32F9D2     NO:   LD   (CVEL),A
D33B F1               POP  AF
D33C C9               RET
                      END
```

LISTADO DE ETIQUETAS

```
ACTIVA  D2F0  CVEL  D2F9  DESACT  D30F  LUCE  D317
NO      D338  POSI  D2F8  VEL1    D2F6  VEL2  D2F7
```



```
10 ' CAP INTERMITENTE
20 '
30 ' CARGADOR BASIC
40 '
50 RESTORE 9000
60 S=0
70 FOR A=&HD2F0 TO &HD33C
80 READ B
90 S=S+B
100 POKE A,B
110 NEXT
120 IF S<>11179 THEN PRINT"ERROR EN DATA
S":END
130 CLS
140 PRINT "PULSA SPACE PARA GRABARLO"
150 IF STRIG(0)<>-1 THEN 150
160 BSAVE "CAS:CAPINT",&HD2F0,&HD33C
170 END
 9000 DATA C3,FA,D2,C3,0F,D3,05,05,01,03
,F3,3E,C3,21,17,D3
 9010 DATA DD,21,9F,FD,DD,77,00,DD,75,01
,DD,74,02,FB,C9,F3
 9020 DATA 3E,C9,32,9F,FD,FB,C9,F5,3A,F9
,D2,3D,B7,20,19,3A
 9030 DATA F8,D2,EE,01,32,F8,D2,CD,32,01
,3A,F6,D2,47,3A,F7
 9040 DATA D2,32,F6,D2,78,32,F7,D2,32,F9
,D2,F1,C9
```



mada del S.O., en orden a manejar la interrupción limpiamente.

El S.O. guarda en la pila todos los registros, incluyendo el juego alternativo, antes de llamar al gancho. Por ello podremos utilizar todos los registros libremente. Existe una excepción, el sistema operativo recoge el contenido del registro de estado del VDP y lo sitúa en el acumulador, por lo tanto, tendremos que conservar el contenido del par AF.

Para que el gancho apunte a nuestra rutina, lo que hacemos es incorporar la instrucción de salto JP nnnn del Z-80 en los 3 primeros bytes del gancho, donde nnnn es la dirección de comienzo de la rutina. Esta operación la realiza la rutina ACTIVA en las líneas 19-27.

La rutina llamada durante la interrupción va a ser LUCE, líneas 35-51.

Lo primero que hace es guardar AF como antes indicábamos. Luego decrementa el contador parcial (CVEL) y se comprueba si es cero.

Si no lo es se recupera AF y se regresa. En el caso afirmativo, se cambia el indicador de estado de encendido-apagado contenido en POSI, y se llama a la rutina de la ROM que nos fija este estado en el piloto. Esta rutina comienza en 0132H y acepta un valor contenido en el acumulador, que si es cero apaga el piloto y distinto de cero lo enciende.

La siguiente operación, es intercambiar los contadores de pausa y destello, y cargar el contador parcial con el contenido de Vel2.

Debido a que estos datos están intercambiándose constantemente, es necesario fijar su valor antes de activar la interrupción, ya que en caso contrario no sabríamos cual es cual más que mirándo el contenido de POSI, coincidiendo cuando este valor fuese cero.

Una vez hecho esto se recupera AF y se regresa de la llamada.

Para desactivar la rutina, lo que deberemos hacer es restaurar la situación inicial del gancho, cambiando el primer byte por la instrucción RET (201). Esto es lo que hace la rutina DESACT de las líneas 29-33.



GUSANO GLOTON

Has de conducir a Tomasín, el gusano glotón, para que se coma todos los caramelos de la habitación sin estrellarse contra las paredes ni contra sí mismo.

Para moverlo dispones de las teclas del cursor. Gusano Glotón es un juego donde la acción es un elemento fundamental, y para que exista acción hemos de hacer que nuestro programa corra a la máxima velocidad posible, por lo cual se ha simplificado al máximo el bucle principal.

COMENTARIO DEL PROGRAMA

1000-1130 Presentación. Se apagan los sonidos y se ocultan las teclas de función. A continuación se fija el modo gráfico multicolor y se fijan los colores correspondientes. En la línea 1060 abrimos el canal hacia la pantalla gráfica, esto es necesario para imprimir téxto. Imprimimos el título, cerramos el fichero, se adorna con un par de líneas horizontales y tocamos una corta melodía antes de empezar.

1140-1340. Entramos ya en lo que es el juego. Se acondiciona la pantalla (esta vez vamos a emplear el SCREEN 1), se introducen los datos de los gráficos directamente en la VRAM y les damos color. Inmediatamente después se inicializan las variables de los puntos y las vidas. Se define una función que tomando como parámetros las coordenadas x e y, nos devuelve el código del carácter que ocupa esa posición en la pantalla. En las líneas 1280-1310 se dibuja el escenario. En las líneas siguientes se inicializan las variables de las coordenadas, la de dirección (S) y el contador

```
1000 ' ****************
       *              *
       * GUSANO GLOTON *
       *              *
       ****************
1010 '
1020 '
1030 '
1040 KEYOFF:BEEP
1050 SCREEN 3:COLOR 15,0,0:CLS
1060 OPEN "GRP:"AS#1
1070 PSET (40,50),1:COLOR 9:PRINT#1,"
GUSANO"
1080 PSET (40,90),1:COLOR 9:PRINT#1,"
GLOTON"
1090 CLOSE
1100 LINE (0,35)-(255,35),11
1110 LINE (0,125)-(255,125),11
1120 PLAY "V15CDEFGDFECFDFECCC","V150
5CDEFGDFECFDFECCC"
1130 IF PLAY(0)=-1 THEN 1130
1140 ' PREPARACION '
1150 KEYOFF:BEEP
1160 SCREEN 1,1,0
1170 COLOR 15,1,1:CLS
1180 ' GRAFICOS '
1190 DATA 60,126,255,255,255,255,126,
60,0,0,24,60,60,24,0,0
1200 RESTORE 1190
1210 FOR A=1080 TO 1095
1220 READ B
1230 VPOKE A,B
1240 NEXT A
1250 VPOKE 8208,144:VPOKE 8209,32:VPO
KE 8216,160:VPOKE 8217,160:VPOKE 8219
,160
1260 VPOKE 8198,208:VPOKE 8199,208
1270 P=0:V=3:DEFFNH(X,Y)=VPEEK(6146+3
2*X+Y)
1280 CLS:PRINT"└▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒┌ └▒▒▒▒
▒▒▒▒▒▒┌║ PUNTOS:00000 ║║ VIDAS:00 ║┌
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒└ ┌▒▒▒▒▒▒▒▒▒▒└
▒▒▒▒▒▒▒▒▒▒▒▒┌";
1290 FOR A=1 TO 17:PRINT"║ēēēēēēēēēēē
ēēēēēēēēēēēēēēē║";:NEXT A
1300 PRINT"┌▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
▒▒└";
1310 LOCATE 25,1:PRINTUSING"##";V
1320 X=12:Y=14:S=4:C=0
1330 XB=X:YB=Y
1340 SOUND 8,15
```

```
1350 D=STICK(0)
1360 IF D=1 THEN S=1
1370 IF D=3 THEN S=4
1380 IF D=5 THEN S=2
1390 IF D=7 THEN S=3
1400 IF S=1 THEN X=X-1
1410 IF S=2 THEN X=X+1
1420 IF S=3 THEN Y=Y-1
1430 IF S=4 THEN Y=Y+1
1440 G=FNH(X,Y)
1450 IF G<>136 THEN GOSUB 1510
1460 P=P+5:LOCATE 9,1:PRINTUSING"####
#";P
1470 C=C+1:IF C>458 THEN FOR A=1TO3:S
OUND8,15:FORB=0TO255:SOUND0,B:NEXTB:S
OUND8,0:NEXTA:P=P+50:GOTO 1280
1480 SOUND 8,12:FOR A=90TO 250 STEP20
:SOUND 0,A:NEXT A:SOUND 8,0
1490 LOCATE Y,X:PRINT"ç"
1500 GOTO 1350
1510 FOR A=15 TO 0 STEP -1
1520 VPOKE 8208,16*A:SOUND 8,15:FOR B
=250 TO 0 STEP -6:SOUND 0,B:NEXT B:SO
UND 8,0:NEXT A:VPOKE 8208,144
1530 V=V-1:IF V<0 THEN RETURN 1560
1540 LOCATE 25,1:PRINTUSING"##";V
1550 RETURN 1280
1560 BEEP
1570 PLAY "V15O4CDO3BO4E2R4","V15O4EF
DG2R4","V15O4GAGO5C2R4"
1580 IF PLAY(0)=-1 THEN 1580
1590 COLOR 7:CLS
1600 PRINT:PRINT"

"
1610 PRINT:PRINT"

"
1620 VPOKE 8219,208:VPOKE8217,208
1630 LOCATE 0,16:A$="PUNTOS:"+STR$(P)
:GOSUB 1700
1640 A$="OTRA PARTIDA (S/N)":GOSUB 17
00
1650 A$=INKEY$
1660 IF A$="S" OR A$="s" THEN RUN
1670 IF A$="n" OR A$="N" THEN SCREEN
0:FOR A=1 TO 50:A$=INKEY$:NEXT A:END
1680 V=16*(2+INT(RND(1)*13)):VPOKE 82
19,V:VPOKE 8217,V:FOR A=1TO50:NEXT A
1690 GOTO 1650
1700 PRINTTAB(15-LEN(A$)/2);A$:PRINT:
PRINT:RETURN
```

de "caramelos" cogidos.

Por último, antes de pasar al bucle principal, se pone el volumen del canal A a 15.

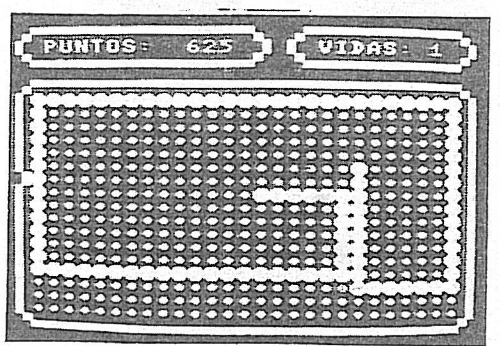
1350-1500 Bucle principal. Se lee el teclado y se guarda en S la nueva dirección. Luego en las líneas 1400-1430 en función del valor de S se incrementan las coordenadas.

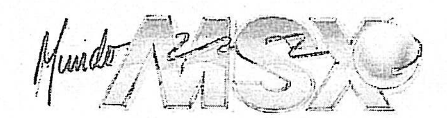
Antes de imprimir tomamos el carácter de la pantalla, y en caso de no ser el 136 (nuestro caramelo), se salta a la línea 1510 donde se rstará una vida. En el caso contrario, se suman 5 puntos y se imprime el marcador, se incrementa el número de caramelos cogidos y se comprueba si se han reunido ya todos. De ser así, hace un sonido, se suman 50 puntos y se empieza de nuevo.

Si no se ha llegado al final, se hace un sonido, se imprime nuestro gusano y se repite el bucle.

1510-1550. Rutina de pérdida de vida. Se cambia el color de nuestro gusano, cambiando en la VRAM el byte que define el color que debe tomar ese carácter en pantalla. Se comprueba si no tenemos vida, si es cero se salta al final. Si no comenzamos de nuevo.

1570-1700. Game Over. Se ejecuta una melodía, dibuja en pantalla GAME OVER y los puntos. Se lee el teclado y al pulsar "S" salta al principio. Si pulsamos "N" se sale del programa. En la línea 1700 está la subrutina que imprime centrados los mensajes de ésta pantalla.





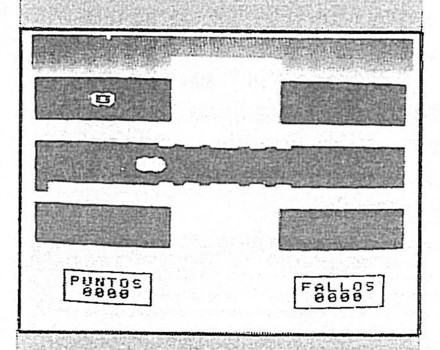
PUENTES

Este programa te ayudará a mantener en forma tus reflejos ya que tendrás que evitar que ningún coche de los que transita por la autopista choque contra el puente móvil que tu manejas, indicando cual es la vía transitable 1, 2 ó 3 según el número que pulses.

EXPLICACION

Entre las líneas 1000 y 1210 se encuentra la pantalla de presentación realizada en SCREEN-3 y en la que podemos apreciar el rótulo "PUENTES" y una melodía (línea 1200).

A partir de la línea 1230 empieza a definirse la pantalla del juego colocándola en SCREEN-1 y dando color (líneas 1260-1280) a los bloques gráficos que compondrán el paisaje que se di-

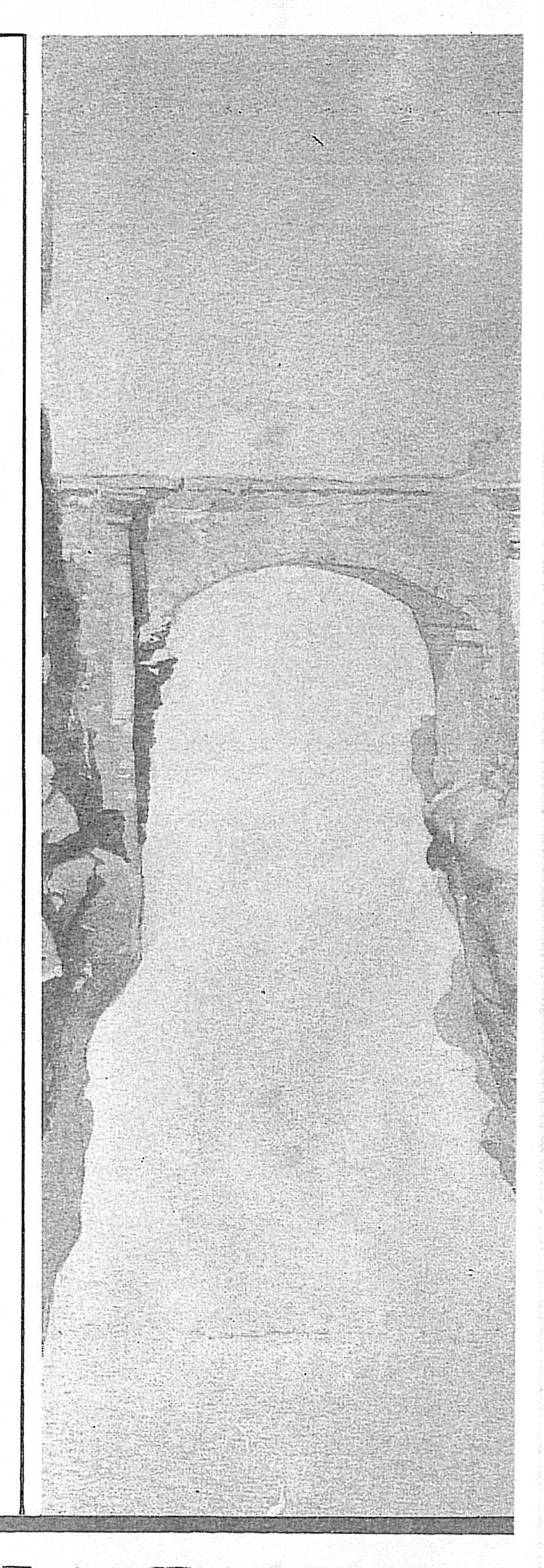


buja entre las líneas 1300 y 1360.

La línea 1370 desvía el programa a la subrutina que genera los SPRITES (líneas 2360-2420).

```
1000 ' **** PUENTES ****
1010 '
1020 '
1030 '
1040 '
1050 KEYOFF
1060 BEEP
1070 SCREEN 3,1,0
1080 COLOR 15,1,1
1090 CLS
1100 OPEN "GRP:" AS#1
1110 LINE (0,120)-(256,60),4,BF
1120 PRESET (39,78),4
1130 PRINT#1,"PUENTE"
1140 PRESET (35,78),4
1150 COLOR 1
1160 PRINT#1,"PUENTE"
1170 LINE (0,60)-(256,60),7
1180 LINE (0,120)-(256,120),7
1190 COLOR 15
1200 PLAY "V15T230GFEDFGEDBAGDFEBCGDF
EBDGEFDCCC"
1210 CLOSE
1220 IF PLAY(0)=-1 THEN 1220
1230 SCREEN 1,2,0
1240 COLOR 1,12,12
1250 CLS
1260 VPOKE 8219,112
1270 VPOKE 8218,17
1280 VPOKE 8216,225
1290 FOR A=1 TO 3
1300 PRINT"
[graphic characters]
";
1310 NEXT A
1320 FOR A=0 TO 19
1330 LOCATE 11,A
1340 PRINT"[graphic characters]"
1350 NEXT A
1360 PRINT"  XWWWWWWY              XWWWWW
WY   VPUNTOSV            VFALLOSV   V O
OOO V            V OOOO V    ZWWWWWW[
        ZWWWWWW[";
1370 GOSUB 2360
1380 P=0
1390 F=0
1400 DEFFNP$(P)=RIGHT$("0000"+MID$(ST
R$(P),2,4),4)
```

```
1410 S=1
1420 GOSUB 1690
1430 M1=0
1440 Y1=0
1450 C1=6
1460 M2=1
1470 Y2=130
1480 C2=4
1490 A=RND(-TIME)
1500 A$=INKEY$
1510 IF S>0 AND A$="1" THEN GOSUB 163
0:S=0:GOSUB 1690
1520 IF S<>1 AND A$="2" THEN GOSUB 16
30:S=1:GOSUB 1690
1530 IF S<2 AND A$="3" THEN GOSUB 163
0:S=2:GOSUB 1690
1540 Y1=Y1+4
1550 IF Y1>256 THEN GOSUB 1750
1560 PUTSPRITE0,(Y1,23+48*M1),C1,0
1570 IF Y1>88ANDY1<168ANDM1<>S THEN G
OSUB 1790
1580 Y2=Y2-4
1590 IF Y2<0 THEN GOSUB 1850
1600 PUTSPRITE1,(Y2,23+48*M2),C2,0
1610 IF Y2>88ANDY2<168ANDM2<>S THEN G
OSUB 1890
1620 GOTO 1500
1630 E=2+6*S
1640 LOCATE 10,E:PRINT"┼███████┼"
1650 LOCATE 10,E+1:PRINT"┼███████┼"
1660 LOCATE 10,E+2:PRINT"┼███████┼"
1670 LOCATE 10,E+3:PRINT"┼███████┼"
1680 RETURN
1690 E=2+6*S
1700 LOCATE 10,E:PRINT"├┼├┼├┼├┼├┼├"
1710 LOCATE 10,E+1:PRINT"┼┼┼┼┼┼┼┼┼┼┼"
1720 LOCATE 10,E+2:PRINT"┼┼┼┼┼┼┼┼┼┼┼"
1730 LOCATE 10,E+3:PRINT"└┴└┴└└┴└┴└"
1740 RETURN
1750 Y1=0
1760 M1=INT(RND(1)*3)
1770 C1=2+INT(RND(1)*13)
1780 GOTO 1950
1790 GOSUB 2040
1800 PUTSPRITE0,,9,1
1810 M1=S
1820 FOR A=1 TO 200
1830 NEXT A
1840 GOTO 1990
1850 Y2=256
1860 M2=INT(RND(1)*3)
1870 C2=2+INT(RND(1)*13)
1880 GOTO 1950
1890 GOSUB 2040
1900 PUTSPRITE1,,9,1
```



PUENTE

Entre las líneas 1380 y la 1490 se inicializan las variables y contadores, dibujandose el primer puente partiendo de una subrutina llamada desde la línea 1420 y que acude a las líneas 1590-1740.

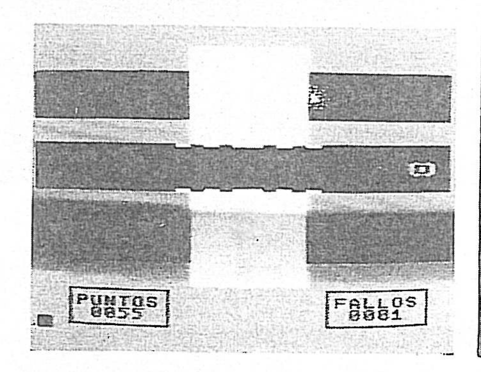
Entre las líneas 1540 y 1620 tenemos el movimiento de los automóviles y el retorno al sondeo de teclas (línea 1620), así como las salidas a las subrutinas que incrementan los puntos o los fallos, en cada uno de los dos coches qeu puede haber en pantalla al mismo tiempo.

Entre las líneas 1950 y 1980 se encuentra la subrutina que incrementa los puntos.

A partir de la línea 1990 y hasta la 2030 tenemos el contador de fallos.

Entre las líneas 2040 y 2100 está la subrutina que genera el sonido de explosión.

A partir de la línea 2110 comienza la rutina de fin de programa con el marcador final y la opción de volver a jugar, esta rutina termina en la línea 2350.




```
1910 M2=S
1920 FOR A=1 TO 200
1930 NEXT A
1940 GOTO 1990
1950 P=P+5
1960 LOCATE 4,22
1970 PRINTFNP$(P)
1980 RETURN
1990 F=F+1
2000 LOCATE 22,22
2010 PRINTFNP$(F)
2020 IF F>9 THEN RETURN 2110
2030 RETURN
2040 SOUND 6,21
2050 SOUND 7,247
2060 SOUND 8,16
2070 SOUND 11,100
2080 SOUND 12,60
2090 SOUND 13,0
2100 RETURN
2110 BEEP
2120 PLAY "V15T200BBBBAGFEDCCCC"
2130 SCREEN 1
2140 LOCATE 0,5
2150 A$="GAME OVER":GOSUB 2300
2160 LOCATE 0,11
2170 A$="PUNTOS: "+FNP$(P)
2180 GOSUB 2300
2190 A$="OTRA PARTIDA (S/N)"
2200 GOSUB 2300
2210 FOR A=1 TO 50
2220 A$=INKEY$
2230 NEXT A
2240 POKE &HFCAB,255
2250 A$=INKEY$
2260 IF A$="S" THEN RUN
2270 IF A$="N" THEN SCREEN 0:END
2280 GOTO 2250
2290 LOCATE 0,0
2300 PRINTTAB(15-LEN(A$)/2);A$
2310 PRINT
2320 PRINT
2330 PRINT
2340 RETURN
2350 END
2360 DATA 0,0,0,1C,7F,F4,F3,F3,F3,F3,
F4,7F,1C,0,0,0,0,0,0,38,FE,2F,CF,CF,C
F,CF,2F,FE,38,0,0,0,8,0,0,24,0,8,3,23
,B,1,83,12,10,84,0,10,0,82,0,88,0,A0,
4,E0,C0,C9,E0,80,8,42,0,90
2370 RESTORE 2360
2380 FOR A=14336 TO 14399
2390 READ B$
2400 VPOKE A,VAL("&H"+B$)
2410 NEXT A
2420 RETURN
```

CUCO

Con este sencillo juego podeís poner a prueba vuestra puntería, se trata de introducir una bola en la boca del pájaro de un reloj de cuco cuando éste la tenga abierta:

EXPLICACION DEL LISTADO

Entre las líneas 1000 y 1210 tenemos la pantalla de presentación realizada en el modo gráfico SCREEN 3, en la que podemos apreiar el rótulo "CUCO" y una pequeña melodía inicial.

Entre las líneas 1220 y 1270 comienza el juego en sí, colocando la pantalla en el modo gráfico SCREEN 2 e inicializando algunas variables así como poniendo a ceros el marcador (línea 1270).

La línea 1280 llama a una subrutina que comienza en la línea 1770 y concluye en la 1870. Esta subrutina es la encargada de dibujar el paisaje y puede ser modificada a vuestro gusto.

La línea 1290 desvía el programa hacia la subrutina que se encarga de definir la forma de los sprites y que se encuentra entre las líneas 2120 y 2190.

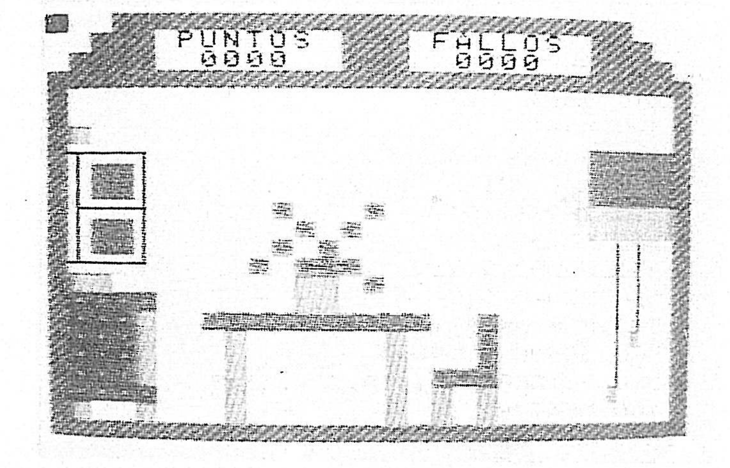
Las líneas 1300 y 1310 inicializan la variable "M" y bifurcan a las líneas 1730-1760 donde se encuentra una subrutina que de modo aleatorio se encarga de, según el valor de "M", colocar al pájaro con la boca abierta o cerrada.



La línea 1330 sondea si se ha pulsado la barra espaciadora. En caso negativo la línea 1340 hace que el programa continúe esperando a que se produzca dicha pulsación.

A partir de la línea 1350 comienza una rutina que, una vez ha sido pulsada la barra espaciadora, desplaza la bola hasta el pájaro.

La línea 1420 comprueba si éste tiene la boca abierta, en cuyo caso hace saltar el programa a la línea 1520.



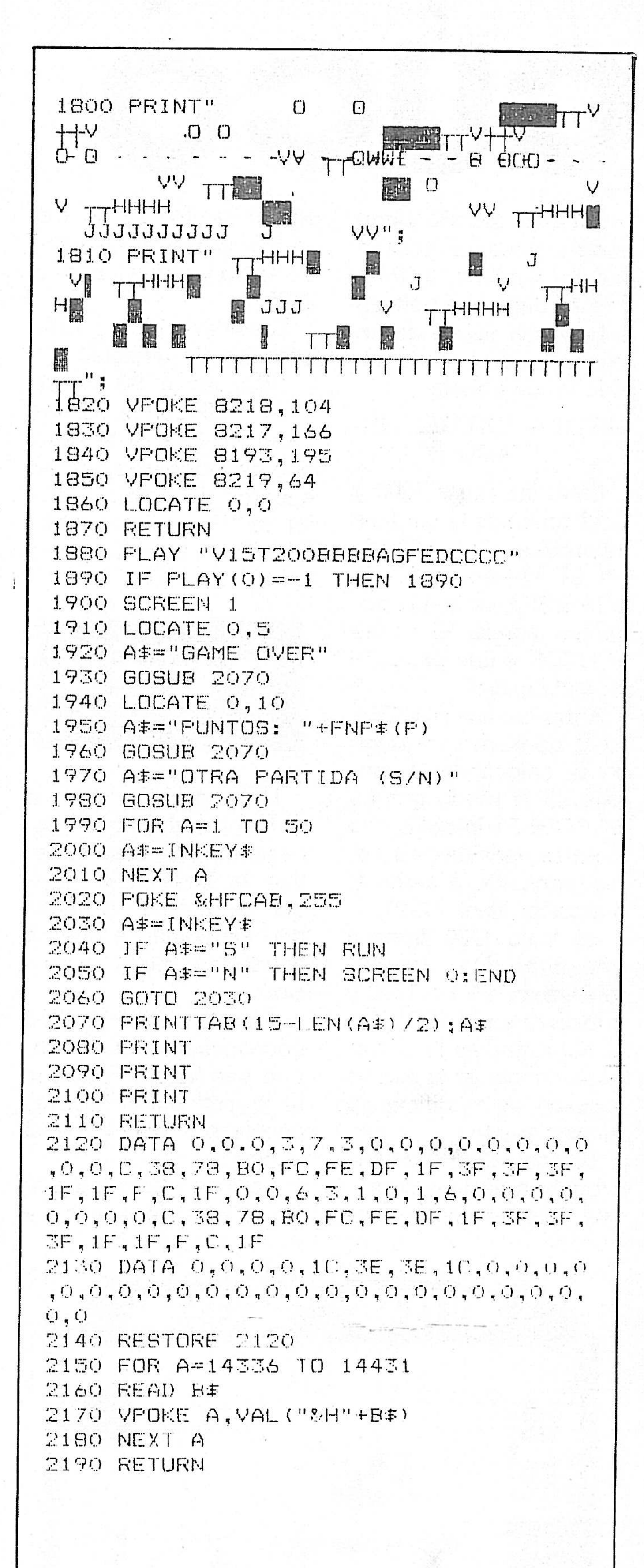
Las líneas 1430 hasta la 1510 entran en acción cuando la bola golpea el pico cerrado de nuestro cuco haciéndola caer al suelo.

Entre las líneas 1520 a la 1680 se encuentra la rutina de "aciertos" que entra en juego cada vez que introducimos una bola en la boca del cuco.

Las líneas entre la 1690 y la 1720 hacen una pausa y una posterior comprobación del número de fallos, que en caso de ser mayor que cuatro, la línea 1710 desviaría el programa a la línea 1880 a partir de la cual y hasta la 2110 se encuentran los mensajes de fin de partida y la opción de volver a jugar.

```
1000 ' *** CUCO ***
1010 '
1020 '
1030 '
1040 '
1050 KEYOFF
1060 BEEP
1070 SCREEN 3,2,0
1080 COLOR 1,15,15
1090 CLS
1100 OPEN "GRP:"AS#1
1110 LINE (50,50)-(210,150),1,BF
1120 PRESET (76,86),1
1130 COLOR 7
1140 PRINT#1,"CUCO"
1150 PRESET(72,86),1
1160 COLOR 4
1170 PRINT#1,"CUCO"
1180 LINE (50,50)-(210,150),6,B
1190 CLOSE
1200 PLAY "V15EDCDEFFDGEFCBAGCFDBEGCC
CC"
1210 IF PLAY(0)=-1 THEN 1210
1220 SCREEN 1,2,0
1230 COLOR 1,15,15
1240 CLS
1250 P=0
1260 F=0
1270 DEFFNP$(P)=RIGHT$("0000"+MID$(ST-
R$(P),2,4),4)
```

```
1280 GOSUB 1770
1290 GOSUB 2120
1300 M=1
1310 GOSUB 1730
1320 PUTSPRITE 1,(52,72),13,2
1330 IF STRIG(0)=-1 THEN GOTO 1350
1340 GOTO 1310
1350 SOUND 8,15
1360 SOUND 1,0
1370 FOR A=52 TO 190
1380 SOUND 0,190-A
1390 PUTSPRITE 1,(A,72),13,2
1400 NEXT A
1410 SOUND 8,0
1420 IF M=2 THEN 1520
1430 SOUND 8,15
1440 FOR A=72 TO 167
1450 SOUND 0,A
1460 PUTSPRITE 1,(190,A),13,2
1470 NEXT A
1480 SOUND 8,0
1490 F=F+1
1500 LOCATE 18,2:PRINTFNP$(F)
1510 GOTO 1690
1520 SOUND 8,15
1530 FOR A=190 TO 198
1540 SOUND 0,255-(A-189)*20
1550 PUTSPRITE 1,(A,72),13,2
1560 FOR B=1 TO 30
1570 NEXT B
1580 NEXT A
1590 SOUND 8,0
1600 FOR A=1 TO 9
1610 M=3-M
1620 GOSUB 1740
1630 FOR B=1 TO 100
1640 NEXT B
1650 NEXT A
1660 P=P+5
1670 LOCATE 7,2
1680 PRINTFNP$(P)
1690 FOR A=1 TO 1000
1700 NEXT A
1710 IF F>4 THEN 1880
1720 GOTO 1300
1730 IF RND(1)>.9 THEN M=3-M:SOUND 9,
15:SOUND 9,0
1740 IF M=1 THEN PUTSPRITE0,(192,73),
5,0
1750 IF M=2 THEN PUTSPRITE0,(192,73),
5,1
1760 RETURN
1770 CLS
1780 PRINT"      TTTTTTTTTTTTTTTTTTTTTTTT
   TTT  PUNTOS  TTT  FALLOS  TTT    TTTT
 0000   TTT   0000   TTTT"
1790 PRINT"TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTT                            TT
                    TT█
             TTRWWY
  ++++TTV++V                   ++++
TV++V                       ++++TTTWWS";
```



```
1800 PRINT"      O    O          █TTV
++V      .O O       █TTV++V
O-O - - - - - -VV  TTQWWE - - B OOO - - -
    VV  TT█   █    █  O          V
V TTHHHH      █           VV  TTHHH█
 JJJJJJJJJ   J           VV";
1810 PRINT" TTHHH█   █     █   J
 V█  TTHHH█   █    █   J    V   TTHH
H█  █      █ JJJ     V   TTHHHH     █
  █ █ █      █    TT█   █    █      █ █
█        TTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TT";
1820 VPOKE 8218,104
1830 VPOKE 8217,166
1840 VPOKE 8193,195
1850 VPOKE 8219,64
1860 LOCATE 0,0
1870 RETURN
1880 PLAY "V15T200BBBBAGFEDCCCC"
1890 IF PLAY(0)=-1 THEN 1890
1900 SCREEN 1
1910 LOCATE 0,5
1920 A$="GAME OVER"
1930 GOSUB 2070
1940 LOCATE 0,10
1950 A$="PUNTOS:  "+FNP$(P)
1960 GOSUB 2070
1970 A$="OTRA PARTIDA (S/N)"
1980 GOSUB 2070
1990 FOR A=1 TO 50
2000 A$=INKEY$
2010 NEXT A
2020 POKE &HFCAB,255
2030 A$=INKEY$
2040 IF A$="S" THEN RUN
2050 IF A$="N" THEN SCREEN 0:END
2060 GOTO 2030
2070 PRINTTAB(15-LEN(A$)/2);A$
2080 PRINT
2090 PRINT
2100 PRINT
2110 RETURN
2120 DATA 0,0,0,3,7,3,0,0,0,0,0,0,0,0
,0,0,C,38,78,B0,FC,FE,DF,1F,3F,3F,3F,
1F,1F,F,C,1F,0,0,6,3,1,0,1,6,0,0,0,0,
0,0,0,0,C,38,78,B0,FC,FE,DF,1F,3F,3F,
3F,1F,1F,F,C,1F
2130 DATA 0,0,0,0,1C,3E,3E,1C,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0
2140 RESTORE 2120
2150 FOR A=14336 TO 14431
2160 READ B$
2170 VPOKE A,VAL("&H"+B$)
2180 NEXT A
2190 RETURN
```



DRIVER

Eres un famoso piloto de RALLY y has sido el encargado de probar el nuevo LOTUS 435-S22-Turbo-Inyección.

El circuito de pruebas es una sinuosa carretera donde, en más de una curva, tu potente coche chillará ruedas.

Los controles del vehículo están a cargo de los cursores de tu teclado:

CURSOR IZQUIERDA.- Gira el volante a la izquierda.

CURSOR DERECHA.- Gira el volante a la derecha.

COMENTARIO DEL PROGRAMA

En este programa se ha aprove chado el scroll que realiza la pantalla del MSX al imprimir en la última línea, para reproducir el desplazamiento de una carretera. Nuestro vehículo es un Sprite de 16x16 pixels, que será destruido en caso de colisionar con los extremos de la carretera.

1000-1210. Presentación. Ha sido realizada en SCREEN 3 (Modo Multicolor), para aprovechar el mayor tamaño de las letras al imprimirse. Se abre el canal 1 para imprimir en la pantalla gráfica. Recua-

```
1000 ' **********
       *        *
       * DRIVER *
       *        *
       **********
1010 '
1020 '
1030 '
1040 '
1050 KEYOFF
1060 BEEP
1070 SCREEN 3
1080 COLOR 15,1,1
1090 CLS
1100 OPEN "GRP:"AS#1
1110 PRESET (36,80)
1120 COLOR 7
1130 PRINT#1,"DRIVER"
1140 PRESET(32,80)
1150 COLOR 4
1160 PRINT#1,"DRIVER"
1170 LINE (0,50)-(256,60),6,BF
1180 LINE (0,130)-(256,140),6,BF
1190 PLAY "V15CEDGFEDFEBGFEDCCCV10CV5
C","V15O3CEDGFEDFEBGFEDCCCV14CV13CV12
CV10CV8CV5C"
1200 IF PLAY(0)=-1 THEN 1200
1210 CLOSE
1220 A=RND(-TIME)
1230 V=RND(1)/10
1240 C=13.94
1250 D=3
1260 IS=0
1270 S=144
1280 DEFFNG=VPEEK(6304+INT(S/8))
1290 DEFFNH=VPEEK(6305+INT(S/8))
1300 SCREEN 1,2,0
1310 COLOR 1,2,2
1320 FOR A=0 TO 22:PRINTTAB(11);"├████████
████┬":NEXT A
1330 GOSUB 1900
1340 SOUND 4,24
1350 SOUND 6,32
1360 SOUND 7,3
1370 SOUND 8,2
1380 SOUND 9,2
1390 SOUND 10,16
1400 SOUND 11,58
1410 SOUND 12,0
1420 SOUND13,14
1430 FOR A=-20 TO 32
1440 SOUND11,90-A
1450 PUT SPRITE1,(141,A),6,0
1460 FOR B=1 TO 75
1470 NEXT B
1480 NEXT A
1490 TIME=0
1500 IF D=1 THEN C=C+V
1510 IF D=2 THEN C=C-V
1520 IF RND(1)>.97 THEN D=1+INT(RND(1
)*3):V=RND(1)/6
1530 LOCATE11+8*COS(C),23:PRINT"├████████
████┬"
1540 J=STICK(0)
1550 IF J=3 THENIS=IS+2
1560 IF J=7 THENIS=IS-2
1570 S=S+IS
1580 IF IS<>0 THEN IS=IS-SGN(IS)
1590 PUTSPRITE1,(S-3,32),6,0
```

dramos el título con 2 LINEs y tocamos una breve melodía, retenida por la función PLAY de la línea 1200 hasta que sea ejecutada totalmente, momento en el cual cerramos el fichero que habíamos abierto (realmente se cierran todos los que hubiese abiertos) para dar paso al juego.

1220-1490. Inicialización. En este amplio bloque se realizan las siguientes operaciones:

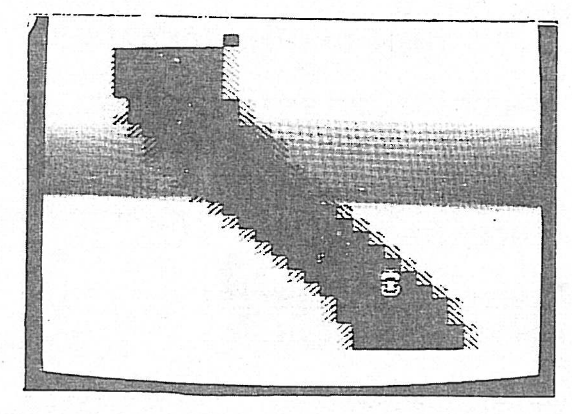
— Fijar el RND con un inicio aleatorio.
— Se inicializan las variables.
— Se definen dos funciones que devuelvan el carácter de la pantalla sobre el que se sitúa nuestro coche.
— Se fija el MODO de texto de 32 columnas, con Sprites de 16x16 y sin CLIC en el teclado.
— Se dan los colores correspondientes al papel, borde y tinta.
— Se dibuja el primer tramo de carreteras (recto).
— Se definen los Sprites.
— Se baja el coche desde el exterior de la pantalla hasta la zona donde permanecerá.
— Se pone el tiempo a cero.

1500-1610. Bucle del juego. Primero se mueve la carretera, se imprime la última línea. A continuación, se lee el teclado para mover nuestro coche. Se sitúa el sprite y se comprueba si choca analizando los valores que devuelven las dos funciones antes mencionadas.

1620-1890. Rutina de choque. Se realiza un sonido de explosión y se imprime el clásico mensaje de GAME OVER. El POKE de la línea 1850 activa las mayúsculas.

1900-1980. Subrutina que mete los datos de los sprites en la VRAM.

1990-2020. Subrutina que imprime el texto centrado.



```
1600 IF FNG<>219 OR FNH<>219 THEN GOT
O 1620
1610 GOTO 1500
1620 BEEP
1630 SOUND 6,21
1640 SOUND 7,247
1650 SOUND 8,16
1660 SOUND 11,100
1670 SOUND 12,60
1680 SOUND 13,0
1690 FOR A=1 TO 5
1700 PUTSPRITE0,(S-3,32),15,A
1710 FOR B=1 TO200
1720 NEXT B
1730 NEXT A
1740 SCREEN 1:P=TIME
1750 SCREEN 1
1760 COLOR 7,1,1
1770 LOCATE 0,6
1780 A$="GAME OVER"
1790 GOSUB 1990
1800 LOCATE 0,12
1810 A$="PUNTOS:"+STR$(P)
1820 GOSUB 1990
1830 A$="OTRA PARTIDA (S/N)"
1840 GOSUB 1990
1850 POKE &HFCAB,255
1860 A$=INKEY$
1870 IF A$="S" THEN RUN
1880 IF A$="N" THEN FOR A=1 TO 50:A$=
INKEY$:NEXT A:SCREEN 0:END
1890 GOTO 1860
1900 DATA 7,F,1C,1B,1F,1B,B,F,B,A,1C,
1F,1F,F,9,7,E0,F0,38,D8,F8,D8,D0,F0,D
0,50,38,F8,F8,F0,90,E0,0,81,11,29,18,
17,4B,D,F7,3A,2D,8F,1B,1E,34,38,20,82
,80,4E,DC,F8,F2,E0,F8,DE,FF,F0,78,39,
28,9C
1910 DATA 0,20,2,10,5,0,52,8,1,20,5,1
0,4,0,12,0,0,40,10,0,40,A,20,40,8,0,4
4,0,90,2,40,10,0,0,40,2,0,10,2,0,40,4
,0,2,10,0,1,0,0,2,20,0,0,48,0,0,9,0,4
,40,0,10,0,4
1920 DATA 1,0,0,2,80,0,0,0,20,0,0,0
,0,1,10,0,0,22,0,0,0,8,0,1,0,0,0,4,0,
0,8
1930 RESTORE 1900
1940 FOR A=14336 TO 14495
1950 READ B$
1960 VPOKE A,VAL("&H"+B$)
1970 NEXT A
1980 RETURN
1990 PRINTTAB(15-LEN(A$)/2);A$
2000 PRINT
2010 PRINT
2020 RETURN
```

LIBRERIA



LENGUAJE MAQUINA PARA MSX

Autor: Joe Pritchard
Editorial: Anaya Multimedia

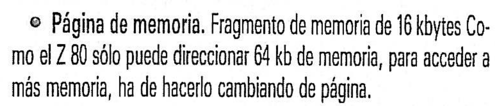
El objetivo de este libro es dotar al usuario de MSX de un manual de programación del Z-80 para que pueda escribir rutinas que hagan uso de varios componentes del sistema. Es un libro interesante para todos aquellos que quieran saber como aplicar sus actuales conocimientos de programación del Z-80 al ordenador MSX en este lenguaje.

Algunos de los temas que trata son:

Código máquina frente al Basic. Los registros en la práctica. Cálculos con 8 bits. Transferencia de datos de 16 bits. Bucles, saltos y operaciones con bloques. Ilustraciones de entrada y salida. El VDP. El PSG.

JERGA INFORMATICA

Glosario de Términos



◦ **Página de memoria.** Fragmento de memoria de 16 kbytes Como el Z 80 sólo puede direccionar 64 kb de memoria, para acceder a más memoria, ha de hacerlo cambiando de página.

◦ **Octeto de memoria.** Expresión equivalente en castellano a la palabra inglesa "byte". Es un conjunto de 8 bits.

◦ **Memoria de video.** RAM de 16 k que posee el microprocesador de vídeo. Su organización varía según el modo de pantalla.

◦ **Machacar.** Expresión utilizada para describir cuándo una zona de memoria crece e invade otra zona alterando su contenido.

◦ **Ensamblador.** Programa mediante el cual a partir de un listado fuente (listado ensamblador), genera un código objeto, comúnmente llamado código máquina.

◦ **CHECKSUM.** (Suma de chequeo). Operación mediante la cual es posible comprobar si un conjunto de datos han sido introducidos correctamente (por ejemplo al copiar un listado), comprobando si coincide la suma de los datos originales, con la de los datos copiados.

◦ **BUFFER.** Zona de memoria, normalmente de corta longitud, que se utiliza para almacenar temporalmente los datos en operaciones de transferencia.

◦ **POKEar.** Introducir un dato en una posición de memoria con el empleo de la instrucción POKE.

◦ **Llamar a una rutina.** En lenguaje ensamblador equivale a hacer:

CALL dirección de comienzo de la rutina y en BASIC sería:

DEFUSR = dirección de comienzo en la rutina: A = USR (0)

◦ **PPJ 8255.** (Interfaz programable de periféricos INTEL 8255). Circuito de E/S muy potente, que en los MSX controla el teclado, la lámpara de mayúsculas, las funciones del cassette, la selección de páginas de memoria, y un puerto de sonido de 1 bit.

◦ **Ranura.** Banco de memoria de 64 kb, que puede funcionar como ROM, como RAM, como conexión de cartucho o como bus de expansión.

◦ **Bus de expansión.** Conector desde el cual se acceden a las funciones del microprocesador. Se usa por ello para conectar periféricos.

◦ **Cascar.** Se dice que el sistema "casca", cuando al producirse un error de cualquier tipo, el sistema operativo pierde el control de sus funciones.

◦ **Gancho.** Conjunto de 5 bytes de RAM inicializados para contener la instrucción RET del z-80 (C9H). Nos permiten redirigir las llamadas (CALL) insertando una instrucción JP nnnn de Z-80, en los 3 primeros bytes del gancho.

FE DE ERRATA

CARGADOR KEOPS

En el número anterior publicábamos el programa Keops sin el programa cargador. Esperamos que nos disculpéis este lamentable error. Y a continuación os ofrecemos el citado programa, recordándoos que hay que teclearlo antes del programa principal y grabarlo en cinta con: SAVE "CAS:KEOPS".

```
10 SCREEN 1:KEYOFF:WIDTH 29:COLOR 2,0,0:
LOCATE 7,9:PRINT"CARGANDO: KEOPS"
20 RESTORE 9000:D=58000!
30 READ A$:FOR A=1 TO 63 STEP 2
40 B$="&H"+MID$(A$,A,2)
50 POKE D,VAL(B$):D=D+1:IF D>58461! THEN
 BSAVE "CAS:C/M",58000!,58461!:GOTO 70
60 NEXT:GOTO 30
70 BLOAD "CAS:",R
 9000 DATA 010008110038219DE2CD5C00C9030
70507036S6E3F170503030 20E1E00C0E0A0E0
 9010 DATA C0A070F8EEA6C0C04040707803070
50703050E1F7765030302020E1EC0E0A0E0C0
 9020 DATA A676FCE8A0C0C04070780000000000
0000000703F70000000000000000000000000
 9030 DATA 0070FC7000000000000000000000000
000000E3F0E00000000000000000000000000
 9040 DATA 0EFC0E0000000000000184945273EF
F7155F17F5FF31F2E4509902224E8FBFC8AAF
 9050 DATA 8EFBAAFC8AF248400944241 7DF3F5
1F571DF553F514F12021892A2E47CFF8EAA8F
 9060 DATA FEFACFF874A2900402011F3161677
57F7F78381E010D0C204080F88C86E6AEFEFE
 9070 DATA 1E1C7880B030000E011F316167757
F7F78381F0C0D01007080F88C86E6AEFEFE1E
 9080 DATA 1CF830B080000814020327 3D270B0
50A30300000000010284 0C0E4BCE4D0A0500C
 9090 DATA 0C00000008040202036 71D670B050
A101818000010204040C0E6B8DCA0A0500818
 9100 DATA 1800000000120002280390052301 1
2002A0002004000900 4200AC088C0022800A0
 9110 DATA 0840000000000080100140000 02480
000040000000001000004004002008008 00210
 9120 DATA 00070F3E1F3F3F262A262A3F3F3F3
F3F3FC0E0E0F0F8F898A898B8F8F8F8F8F8
 9130 DATA 00000000000000000000000000000
00000000000000000000000000000000000
```



RUTINAS UTILES DE SONIDO

Simulación del romper de las olas en un acantilado

```
10 SOUND 7,&B110111
20 SOUND 12,50:SOUND 13,14
30 SOUND 6,10:SOUND 8,16
```

Sonido de alarma utilizando únicamente el segundo canal

```
10 SOUND 7,&B111101
20 SOUND 12,5:SOUND 13,14
30 SOUND 2,170:SOUND 9,16
```

Simulación de un disparo de cañón seguido de una posterior explosión

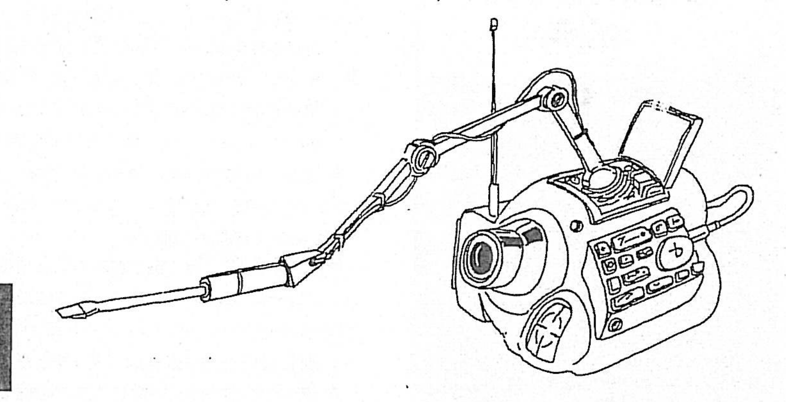
```
10 SOUND 7,&B110111
20 SOUND 6,30:SOUND 8,16
30 SOUND 12,150:SOUND 13,11
```

Simulación del sonido emitido por un helicóptero

```
10 FOR I=0 TO 13:READ S:SOUND I,S:NEXT
20 DATA 5,5,0,1,24,0,9,3,2,2,16,90,2,12
```

O lo que es lo mismo

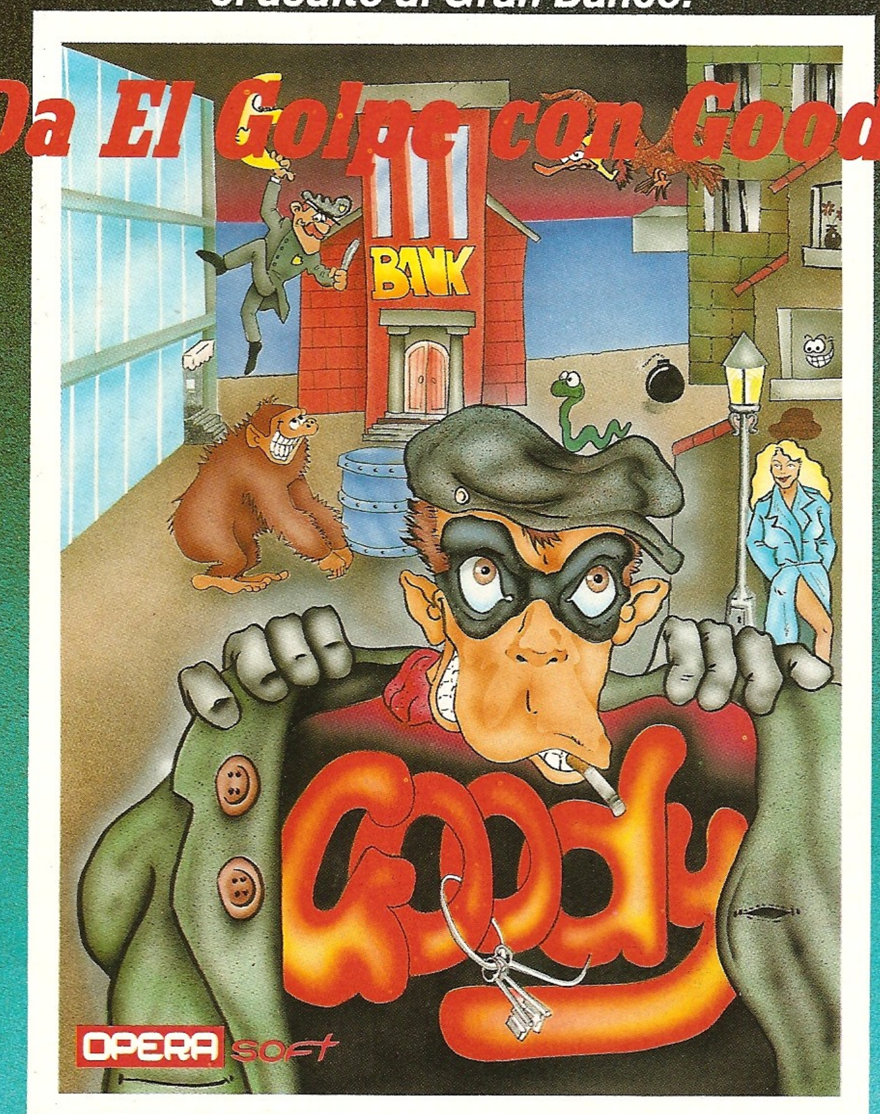
```
10 SOUND 0,5:SOUND 1,5:SOUND 2,0
20 SOUND 3,1:SOUND 4,24:SOUND 5,0
30 SOUND 6,9:SOUND 7,&B000011:SOUND 8,2
40 SOUND 9,2:SOUND 10,16:SOUND 11,90
50 SOUND 12,2:SOUND 13,12
```



que no digan que no das ni golpe

Goody tiene un buen plan, pero los grandes planes nunca son sencillos, por eso necesita un buen socio como tú, experimentado y audaz.

Esta es tu oportunidad para dar el golpe del Siglo: el asalto al Gran Banco.



Versión para PC y Compatibles

También para Amstrad, MSX, Spectrum, Spectrum + 3 y Commodore

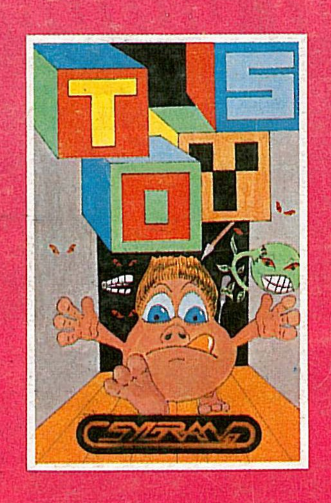


Pza. Santa Catalina de los Donados, 3, 4º Dcha.
28013 Madrid. Tel. 241 92 70 / 241 96 82

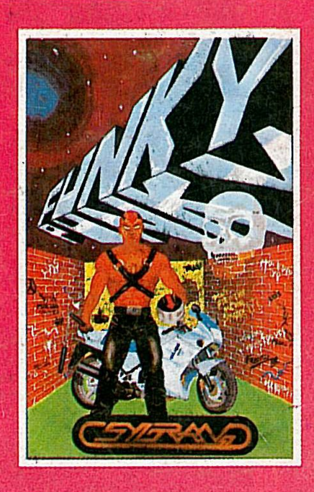
Distribuidor en Cataluña
Discovery Informatic
Telfs.: (93) 256 49 08 - 09

Si no lo encuentras en tu distribuidor habitual llámanos: (91) 241 92 70 - 241 96 82

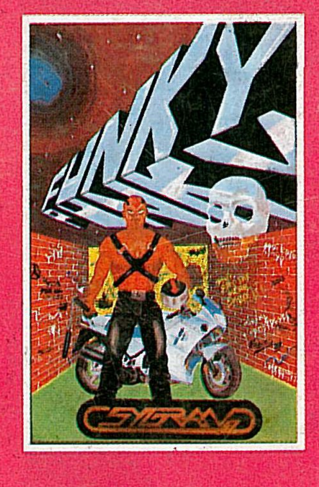
PARA TU...



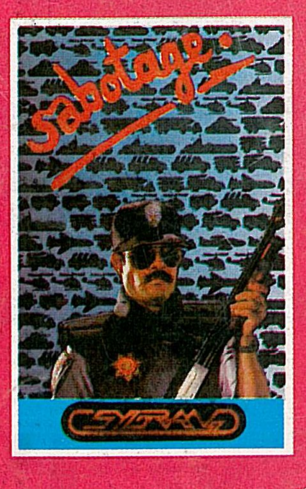
AMSTRAD



SPECTRUM



MSX



MSX

500 pelas